An Interop Publication

CONNEXIONS®

The Interoperability Report

February 1995 Volume 9, No. 2

ConneXions — The Interoperability Report tracks current and emerging standards and technologies within the computer and communications industry.

In this issue:

ConneXions is published monthly by Interop Company, a division of SOFTBANK Exposition and Conference Company, 303 Vintage Park Drive, Foster City, California, 94404–1138, USA.

Phone: +1 (415) 578-6900

Fax: +1 (415) 525-0194

E-mail: `connexions@interop.com`

Subscription hotline: 1-800-575-5717

or +1-502-493-3217

ISSN 0894–5926

From the Editor

Last month we looked at electronic mail, its history, features and even some of its problems. This month, Daniel Blum and David Litwack take us one step beyond e-mail to mail-enabled applications or "Workflow Automation." The article is adapted from their newly published book *The E-Mail Frontier: Emerging Markets and Evolving Technologies*. We hope to have a review of this and many other books in future editions.

By far the fastest growing and most exciting application on the Internet is the World-Wide Web (WWW). Every day, hundreds of new Home Pages are added to the ever growing mesh of hypertext information around the globe. From the user's perspective, the Web is fun and easy to use. New and more powerful Web "browsers" are appearing frequently for all the major computing platforms. The amount of interesting material you can find while "cruising" the Web is growing rapidly. Perhaps the only downside for the user is the amount of time it takes to transfer some of the files that comprise the images or audio/video clips one finds on the Web. The expression "don't try this at home!" came to mind recently as I waited patiently for the transfer of a 60 second audio clip over my V.32 modem. (It took a lot more than 60 seconds to transfer!) But all in the all, the Web is both great fun and one of the most useful things to hit the Internet in a long time.

But what if you decide to become an electronic publisher? How do you go about putting your material on the Web? And once you have it all running, what are some of the performance issues you should know about? These and many other questions are answered in this month's in-depth analysis of Web Servers. The article is by Jon Crowcroft and Mark Handley of University College London (UCL).

Our first NetWorld+Interop of the year takes place March 27–31 in Las Vegas. If you have not done so already you should make your reservation today by calling 1-800-488-2883 or +1 415-578-6900. Two weeks before the show we will be bringing you N+I Online! our Web-based "virtual" extension to the NetWorld+Interop event itself. For more information about N+I Online, see page 29, and don't forget to check it out on March 13th at `http://www.interop.com`.

Included with this edition of *ConneXions* is the 1987–1994 index of back issues. It is hard to believe, but we've now published nearly 100 issues and almost 3,000 pages. All back issues are still available. Your continued input is appreciated as always. Send your questions, comments, suggestions, articles, letters to the Editor, or subscription questions to: `connexions@interop.com`

Workflow Automation and Electronic Commerce

by Daniel J. Blum and David M. Litwack

Introduction

White collar *assembly lines* have existed for some time. One very elaborate such system, which was recently in use at Schiphol Airport in Amsterdam, involves the physical routing of paper in cylinders through pneumatic tubes that terminate at the white collar worker's desktop. (The French PTT also offered a pneumatic tube mail system between post offices throughout Paris until the late 1970s. Eugene Lee of Beyond, Inc. refers to such vacuum systems as "Hoovernets." We are grateful to Mr. Lee for the conceptual framework he provided on mail-enabled applications and workflow automation.)

Such a system could be described as the *plumbing* that enables the transport of objects, provides almost unlimited bandwidth, is relatively secure, and uses existing forms. It lacks such niceties as directories, digital signatures, or other methods of authentication and still requires the rekeying of information. Thus, while the concept of a white collar assembly line (or enterprise work process infrastructure) is not new, the workflow automation tools available today to support such an infrastructure certainly are; and the very availability of these tools serves to suggest new, faster, and more efficient ways of doing business.

Workflow automation and the messaging development platform

Although there has always been something of a paradigm disparity between shared database approaches and the store-and-forward approaches to workflow, electronic messaging remains a key enabling infrastructure for workflow automation.

The shared database approach is most effective when the updating of information can be done in a centralized fashion by a relatively small number of users. When workflow applications become more diffuse, however, involving loosely coupled, relatively unpredictable conjunctions of multiple users, multiple applications, and multiple databases, the use of messaging becomes critical.

Even database-oriented workflow system can make good use of add-on messaging capabilities. E-mail can be used to request action from a user, to notify the user of database changes, and to transmit information to staff members who do not have access to the database. Workflow systems such as Lotus Notes are in fact hybrids between mail-enabled applications and shared database applications.

Workflow systems with messaging components can be based on any of several technical tools and approaches. These include:

- *Smart Applications:* The intelligence is in the application; the messaging network provides simple transport.
- *Smart Mailboxes:* The intelligence to route and process workflows through messaging is in the mailbox.
- *Smart Messages:* The intelligence is in the message. (Many of the concepts and terminology set forth in the remainder of this section owe a great deal to a presentation by Mr. Theodore Myer of Rapport Communication at the 1991 EMA conference.)

Smart applications

Intelligent, distributed messaging applications are not a new phenomenon. Linking these applications using interoperable messaging, however, is new. Most of today's workflow automation, or advanced groupware products fall into the smart application category. Such software solutions coordinate access to a range of services through mail enabling, database sharing, or a combination of the two.

In the sections below, we model a Purchasing Application as a smart application, provide a case study of a smart applications development platform, and consider some of the reasons why workflow applications will ultimately move beyond the smart application model to smart mailboxes and smart messaging.

The key feature of smart applications is their facile and transparent linkage to an embedded messaging utility through APIs or file-based interfaces. Another key feature is the integration of messaging and database resources, including directories.

Suppose a Purchasing Application receives a purchase requisition from an employee (or even another line of business application). It joins the requester name with the type of product and its actual or estimated cost. It opens an approval list and generates e-mail forms to each manager. The managers are thus prompted to review the requisition, fill out the form, and reply. If a manager delays too long, the application dispatches a reminder. The eventual reply (a completed form) does not go to another human user but back to the agent, who circulates it as further e-mail messages to a chain of co-signers prior to feeding the approved information back into the database and proceeding with the purchasing process. Accounts payable and budgetary systems are subsequently activated.

While integrated messaging networks are of critical importance to making our Purchasing Application scenario possible, much of the development effort in producing such an application would necessarily have gone into interfacing with the messaging system, directories, and mail folders. Since few organizations can afford to develop many Purchasing Applications at this level of sophistication from the ground up, a market has emerged for powerful tools to enable the rapid development of smart applications. A market has also emerged for smart (or workflow) application development platforms. High-end products include Beyond Corporation's BeyondMail™, Digital Equipment Corporation's TeamLinks™, Hewlett-Packard's Workflow Manager™, Lotus Development Corporation's Notes™, Reach Software's Workman™, and Novell's GroupWise™ (formerly WordPerfect Office).

Beyond smart applications

As powerful and popular as workflow development platforms like Lotus Notes may be, they remain proprietary applications. Few vendors can or will afford to match the investment required to produce such high end products. Products that compete with Lotus Notes and other industry leaders will find much of their intelligence and processing power absorbed in managing the communications burden. The application or application development platform still has to micromanage the operation of the mail system, sifting mailboxes for replies and generating notices. It still has to define rules and semantics for the routing and circulation of the message and for tracking the status of messages. Each intelligent, distributed application dealing with that same workgroup, department, or enterprise, must duplicate these remaining redundant communications chores. But what if that were not the case?

Economies of scale can be achieved by pushing intelligence down into the messaging network. If intelligent routing and filtering services were available in the messaging platform itself, developers of smart applications and smart applications development platforms could focus exclusively on applications, not communications issues.

continued on next page

Workflow Automation *(continued)*

Given such support, automated workflow system development will be streamlined, utility will increase, and prices will fall. Large numbers of tightly customized and customizable workflow applications would rapidly emerge to support the automation of intra-organizational transaction flows such as the movement of expense reports, time-sheets, purchase requisitions, and human resource forms. Indeed, this process is well underway.

A messaging system can become a development platform by bringing to bear various kinds of smart messaging techniques. Smart messaging can be implemented at the mailbox level, at the message level, or at both levels. Smart mailboxes can filter messages, forward messages, and trigger events. Smart messages can embed procedures to be executed or interpreted before or after message delivery. With both smart mailboxes and smart messages, the messaging system can become an intelligent, programmable instrument able to embody or distribute complex procedures.

Smart mailboxes

Smart or programmable mailboxes add intelligence to the messaging system and offload communications chores from the application. Smart mailboxes can be used to filter messages, forwarding or deleting useless information; they can be used to trigger special procedures; and they can be used to implement complex rules associated with routing, approval, and forms construction. We will discuss smart mailboxes primarily in terms of private electronic commerce, or workflow, since workflow is the environment where smart mailboxes are mostly found. However, smart mailbox functionality can also be built into public messaging networks.

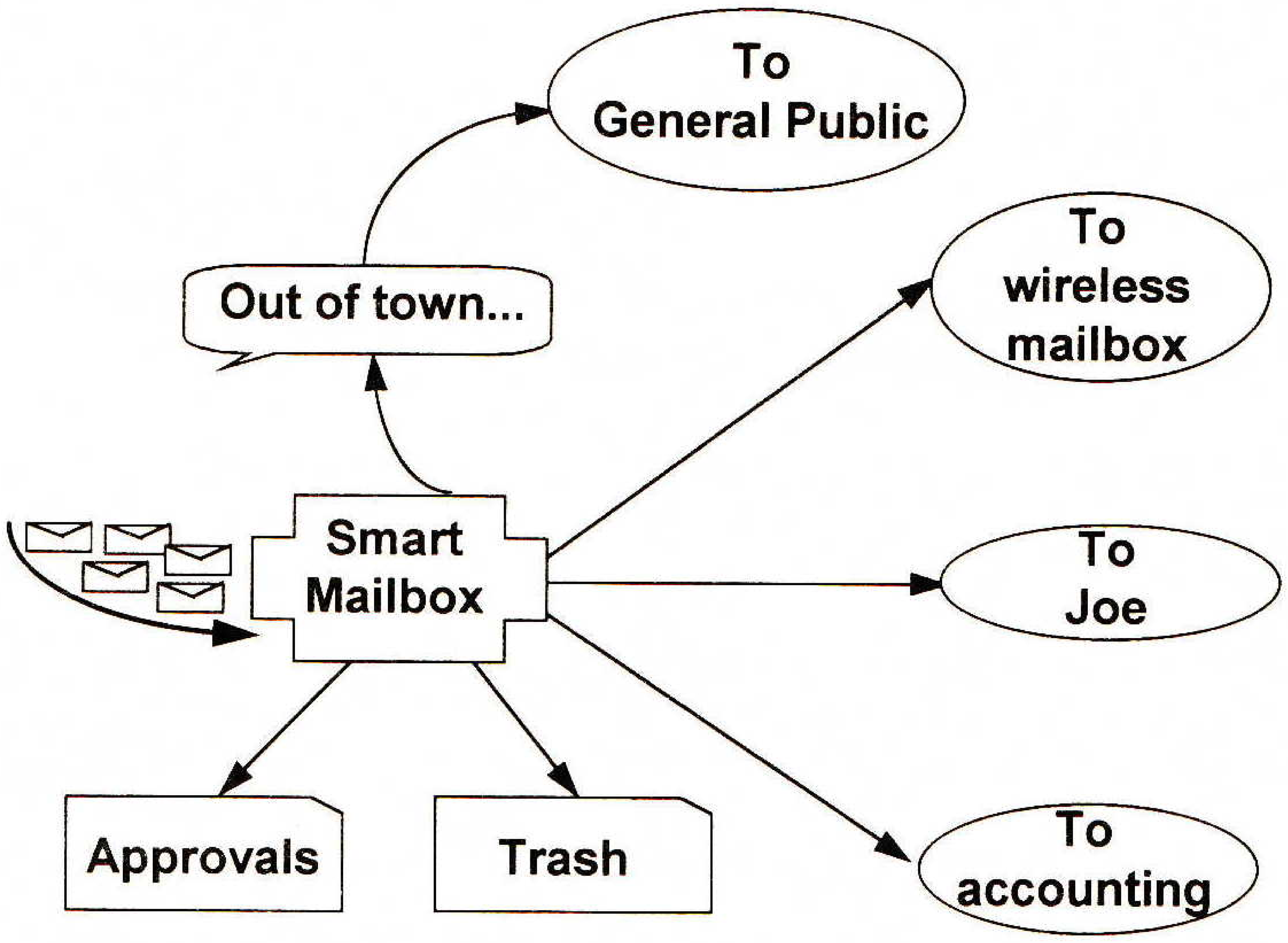


Figure 1: Smart mailbox functionality

Smart mailboxes may be used by people or applications. Some or all of the Purchasing Application's processing described above could have been implemented with the aid of a smart mailbox. People can use these mailboxes for filtering and forwarding; in this regard, the mailbox can alleviate a common complaint from e-mail users that they are inundated by a flood of junk mail. Filtering can select the important messages, delete or forward the unimportant ones, respond automatically to routine requests, and organize information into folders. Examples of filters (and other processes triggered by the mailbox) are suggested below:

- Forward messages from the boss to my wireless mailbox
- Forward messages with Subject line containing the words "Customer Order" to Joe
- File any messages with Subject line "Purchase Requisition Approval" in my approvals folder
- Send spreadsheet file in reply to budget requests from Accounting.
- Reply to any other messages with a note explaining that "I will be out of town until Thursday"
- Put any other messages into the Trash folder if they haven't been read after a week

In general, many products offered today are in fact mail-enabled applications which can be triggered or awakened by the arrival of a certain message at a smart mailbox. Mail-enabled applications may also have the intelligence to manipulate their smart mailboxes through an API. Components of distributed applications can be set up on the messaging network as virtual users with smart mailboxes to perform various kinds of electronic routing-and-approval groupware processes. For example, company employees reporting a casual encounter with a potential customer at a trade show might fill out a "Sales Lead" form and send it to the Sales Department mailbox. The mailbox would file the lead in a database, cross reference it against previous customer leads, prioritize it, assign it to a free salesperson, track contacts with the lead, and maintain statistical records all the way to the point of order processing where inventory, production, and accounting logic would be triggered.

To the degree that smart mailbox functionality is self-contained and driven by recognized characteristics of a message (such as originator, recipients, subject, or priority which are almost universally supported), proprietary smart mailbox implementations could scale well to the multivendor department, enterprise, or vertical industry messaging environment. Forms of smart mailbox processing which are driven by more deeply embedded message information (for example, filling in fields in the message text with database values) scale only by bilateral agreements and could become the subject of future standardization. Other likely topics for future standardization are electronic forms and APIs which applications could use to dynamically manipulate smart mailboxes from multiple vendors.

Smart Folder functionality is a subset of some smart mailbox implementations. Smart folders are aware of the internal attributes of message-borne objects, such as forms. They enable the user to take or define different views of stored messages based on various fields. With a mixture of private folders and shared folders existing alongside many other objects and object collections in the distributed object-oriented operating systems of the late 1990s, e-mail will merge seamlessly into an enterprise database.

Smart messages

In the preceding section, we noted the potential of smart mailboxes to position intelligence at designated static points in the messaging network. However, this approach has a built in scalability limitation. That is, distributing processes across too many smart mailboxes eventually creates excessive complexity even within the internal world of private commerce, let alone in the vast reaches of public commerce.

continued on next page

Workflow Automation *(continued)*

Another approach, which can be combined with smart mailbox technology, utilizes smart messages. This approach involves actually embedding instructions or intelligence in the message and sending that message to selected points in the network. With smart messaging, compiled object code, macros, or scripts are placed inside a message and executed automatically once the message reaches a mailbox. The message may propagate itself and visit multiple stops in support of a process involving loosely coupled network elements cooperating in a particular task. For example, the approvals segment of the Purchasing Application scenario could have been implemented using a smart message that propagated itself through a circulation list of approvers.

Many types of distributed processes can be supported in this fashion. In Figure 2, a user mails a request for a report to his or her expert system. The expert system prepares a smart message and dispatches it to a mail user who is then interviewed using an electronic form. After several additional stops (perhaps to query bulletin boards, news feeds, databases, or spreadsheets) the message comes home to the requester. Such smart messaging could be used for self-activating questionnaires, time card collection, expense report collection, or any procedure requiring remotely triggered program interaction with the user.

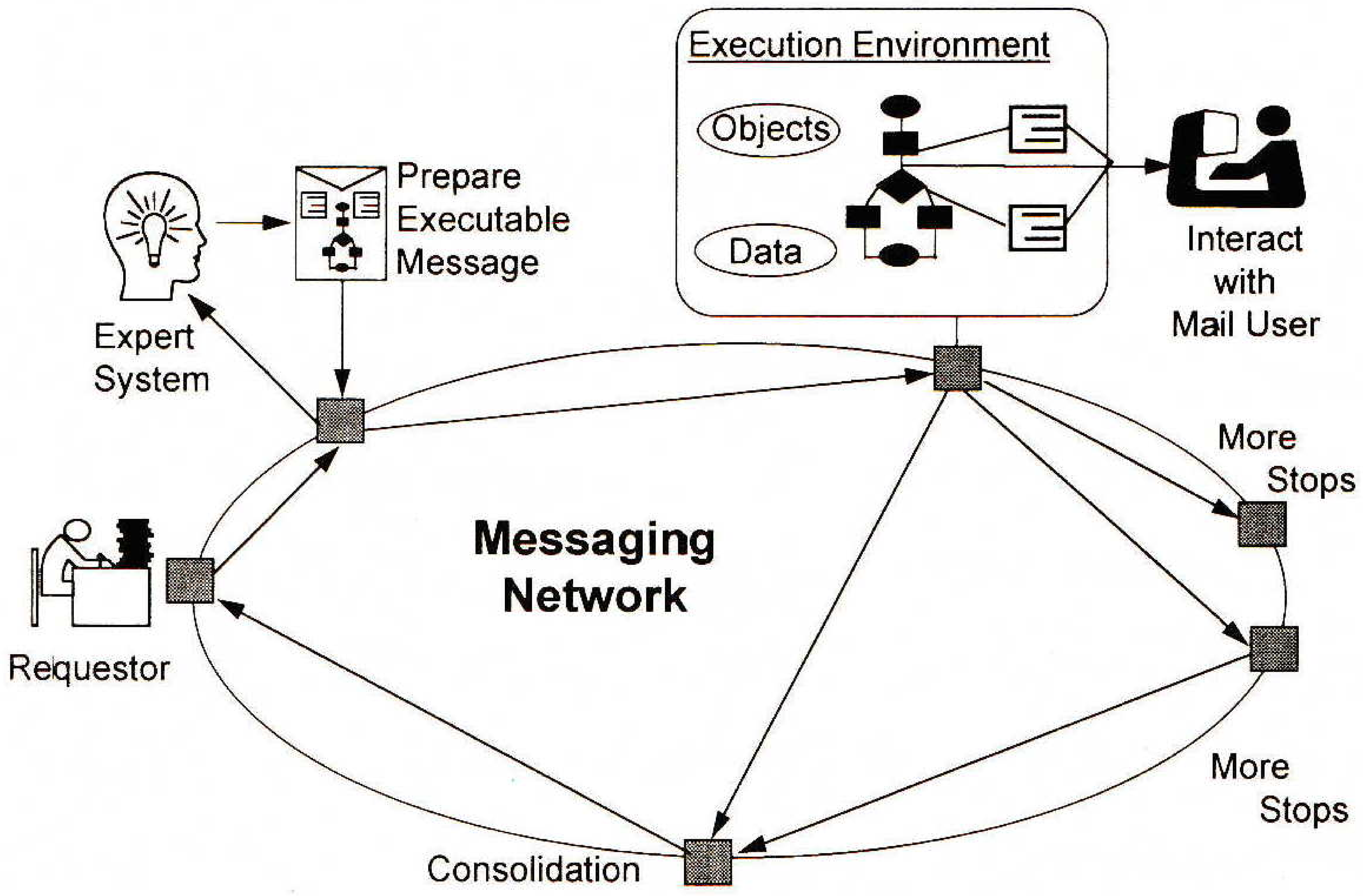


Figure 2: Smart messages

Security

Security will be a major issue for smart messaging technology. Smart messages may require strong authentication of the generating user or application as well as digital signatures to guarantee the integrity of the embedded procedures. Both capabilities could be built on top of 1988 X.400 or the Internet *Privacy Enhanced Mail* (PEM). In addition, access control authorization schemes will be needed to control which data, devices, and other objects the smart message is allowed to use or see. The smart message execution environment will need constraints both to enforce the access controls and to limit the havoc that might be wreaked by a virus designed for this medium. These constraints might be imposed by execution or interpretive shells enforced in the operating system or by placing inherent limitations in the language used for executable messages.

The issue of standards is also important in the smart messaging medium, since one of its main attractions is its potential ability to deliver intelligence to any location reachable by the messaging system. But without standards for the language used, smart messaging will not scale well to the enterprise or vertical industry level. Therefore, we consider two possible standards: a public domain language called *Tcl* and General Magic's *Telescript* language.

Tcl case study

An indication of the viability of smart messaging is the creation of a powerful "universal" scripting language called *Tcl* (pronounced "tickle") and an attendant toolkit called *Tk*. *Tcl* is the acronym for "tool command language" and *Tk* is a "toolkit" for the X Window System.

The intent of *Tcl* [2,11] is to foster a single interpretive language which controls all aspects of a variety of interactive applications. These aspects include the functions, interfaces, and composing pieces of a given application as well as the communication between applications. *Tcl* offers a syntax similar to *sh*, **C**, *Lisp* and generic facilities deriving from built-in commands. *Tk* extends the core *Tcl* facilities by providing commands for building user interfaces via *Tcl* scripts.

Tcl and *Tk* are touted as providing a number of benefits to users and application developers. [In a presentation at the November '93, *E-Mail World* Conference, Nathaniel Borenstein noted, as a credibility check, that he had abandoned all his previous interactive mail languages (*Ness, ATK, ATOMICMAIL*) for *Tcl*—one he did not invent.]

First, they speed up development vis-a-vis using a lower level programming language (less to learn and less code than in writing in **C**, for example). Also, *Tcl* is an interpreted language which means that the developer can generate and execute scripts on the fly (without recompiling or restarting the application). *Tcl* commands, which are linked to a *Tcl* library, can be coded within an application and executed at runtime. Thus, an application for reading an electronic bulletin board might contain a *Tcl* command to query the bulletin board for new messages and another to retrieve a given message. Scripts can then be written for cycling through the new messages and displaying them (or a selected, topical subset) one at a time or recording which messages have been read and which have not in disk files.

Tcl scripts can be used to allow different applications to work together. Since any windowing application based on *Tk* can send a *Tcl* script to any other *Tk* application (to be executed by the recipient), *Tcl* can be used to make multimedia effects more accessible, to enable spreadsheets to update themselves from database applications, and to modify the behavior of live applications as they run. Its proponents also claim that, once an application developer learns *Tcl* and *Tk,* they will be able to write scripts for any *Tcl/Tk* application by learning a few application specific commands.

Tcl and *Tk* can be used specifically for active messages as follows [2]:

- An arriving e-mail message contains the program built using the scripting language.
- The recipient executes the program to read the message.
- The program can then interact with the user—for example, performing a survey or providing a form template.
- The program can take such actions as sending a response or a filled-in form to an authorized recipient.

continued on next page

Workflow Automation *(continued)*

Tcl/Tk also comes with mechanisms to protect calls including access controls, a series of careful checks, encryption techniques, digital signing, and provisions for a certification authority. These mechanisms have become core to a subset of the *Tcl* environment called *Safe-Tcl* which is intended to be safe for evaluating programs potentially written by an unknown or hostile party.

Tcl source, including the *Tcl* command language library, the *Tk* toolkit, and a few *Tcl*-based applications, is currently publicly available. The fact that it has been in the (Internet) public domain and subject to experimentation since 1991 suggests that it has attained a modicum of stability. (*Tcl* is in version 7.2 and *Tk,* in version 3.5.) Meanwhile, a working group, inspired and led by Nathaniel Borenstein and populated by a number of Internet luminaries, was formed in 1993 to implement a prototype *Safe-Tcl.*

Telescript case study

Whereas *Tcl* and *Safe-Tcl* comprise a relatively well-established scripting language that has been extended for smart messaging use, *Telescript* is a new, object-oriented language designed from the beginning with the intent of creating a smart messaging (or more generally, distributed programming) infrastructure. (As of 1994, both the Telescript and *Tcl* languages were too new and untried for us to make any prediction as to which will "win" as a standard. Other smart messaging languages also exist and are under development.)

There are two essential concepts in Telescript: *Agents* and *Places.* Agents are active Telescript programs, either residing within a Place to fulfill a custodial or network management role, or on the move to visit places on behalf of a user. Places are those sites in the network capable of executing Telescript programs. In an electronic messaging environment, mailboxes can be places and smart messages can be agents. In the world of private commerce (as embodied in the internal portion of our Purchasing Application scenario), a purchasing agent could visit a database place to obtain the list of approvals and multiple mailbox places to obtain the approvals.

The General Magic Telescript White Paper [3] predicts that the electronic marketplace will be full of Telescript places. For example, a user's home place might exist in a personal communicator, while other places might exist as shop places nested within a virtual electronic mall place housed on a public service provider's mainframe. Thus, a user might send an agent to the directory place in an electronic mall to obtain a list of shops and services; another agent might visit a ticketing place to purchase theater or sporting tickets.

Agent *travel* between Telescript *regions* (akin to domains) is initiated by means of the Telescript *go* instruction. Parameters to the *go* instruction include the destination, a *telename,* and the agent's *ticket.* The telename identifies the originating user, and the ticket indicates the level of resources the agent is allowed to consume. Once at the destination place, an agent may use the *meet* instruction to interact with another agent. The agent may also establish a *connection* to communicate with another agent back at the home place or another location. Agent transport and connections are accomplished via *Telescript's Platform Interconnect Protocol* (PIP), which is described as a thin veneer over a wide variety of communications media, including e-mail. PIP handles agent procedure, data, and execution state encoding as well as authentication exchanges.

Once in place, Telescript agent programs are interpreted by a Telescript engine. The agent has no direct access to the operating system's resources. A number of security precautions are built into the engine, including the concept of a *permit.* The permit constrains the agent's lifetime, size, and use of resources. The permit is negotiated based on the agent's ticket and the place's policies. In some cases, access may be refused. Access may also be refused if the credentials in the agent's telename cannot be properly authenticated using RSA.

The engine interacts with the platform via three APIs: an External Applications API, a Storage API, and a Transport API. These APIs and the Telescript language will be published by General Magic. The engine is designed for portability. General Magic will make available a *Telescript Developer's Kit,* which provides object code for the engine, as well as tools, documentation and sample programs, and a *Telescript Porting Kit,* which provides source code and documentation. The developer's kit is intended for application developers and the porting kit for system or platform manufacturers.

As of early 1994 there were two existing Telescript engine implementations: one within General Magic's Magic Cap operating system, and the other within AT&T's Personal Link service.

Distributed programming object oriented perspectives

When considering software development problems in the complex world of networked applications, great economies of design and development can be achieved from studying the problem using an object-oriented approach. From the vendor perspective, object-oriented analysis should be applied to as many application scenarios as possible while designing the smart messaging products. From the user perspective, object-oriented analysis could validate the vendor's design against the goals of enterprise development projects.When applied to smart messaging and workflow automation, this type of analysis is still in its infancy and no doubt constitutes a fertile area for academic theses and further research. Here we may find the proving ground where e-mail and artificial intelligence technologies will ultimately converge, where the infrastructure for electronic communities will be constructed, and where workflow automation breakthroughs will be forged.

Smart mailboxes may be viewed as stationary objects which interact with message objects and system objects. What these objects and their interfaces look like is very much a function of the smart mailbox implementation and the implementation of its attached applications. Interfaces may be elegant and flexible or cumbersome and complex. Rules fed into the smart mailboxes can be treated as objects as well.

The executable portion of smart messages may be viewed as *methods* (object-specific procedures) dispatched by local objects to manipulate remote objects. A smart message contains one or more embedded methods able to interact with the objects in its execution environment. These methods may be either standalone programs which interact with objects in the execution environment, or they may be subroutine components which mesh with larger programs prepositioned in the execution environment. A standalone method example would be a program sent to remotely interview a user as part of a survey.

In a more interesting case, the executable portion of an smart message may be part of a distributed method. In other words, the transferred executable element plugs itself in, as a user-defined extension component, to a prepositioned program in a remote environment.

continued on next page

Workflow Automation *(continued)*

For example, a user might send a smart message containing a user-defined method for scanning a bulletin board to locate interesting items. The resulting specialized method would be executed in a highly integrated fashion by a bulletin board server for each bulletin board item available under a topic or list. In this case, the bulletin board is a generic object with a set of generic methods called a bulletin board scanning program and the smart message contains specialized extensions to those generic methods.

In another variation, the smart message may itself be a generic method which plugs into locally specialized methods at the execution scene. These local methods might know how to manipulate a user's printer, terminal, or other components of the local environment.

Smart messages may also extend their environment by defining new object classes or subclasses. Smart messages can extend themselves by discovering and using new classes of information.

Summary

When ultimately in place, the global messaging infrastructure will be positioned to provide a store-and-forward engine for workflow automation and all manner of electronic commerce applications in addition to interpersonal messaging and attached file transfer. These are the breakthrough applications that will change the nature of how business is done—a change deemed essential for enterprises to remain competitive.

Figure 3 diagrams the architectural hierarchy of the electronic applications and infrastructure components that we have discussed in this article. As the figure suggests, many levels of interfaces and platform functionality can and do coexist. Early mail-enabled and workflow applications interface directly to the messaging platform. Simultaneously, smart application development platforms such as Lotus Notes have been enormously successful. More recently, intelligence is being added to the messaging infrastructure itself. Figure 3 suggests that the highest degree of leverage can be obtained by combining technologies at all levels of the architecture.

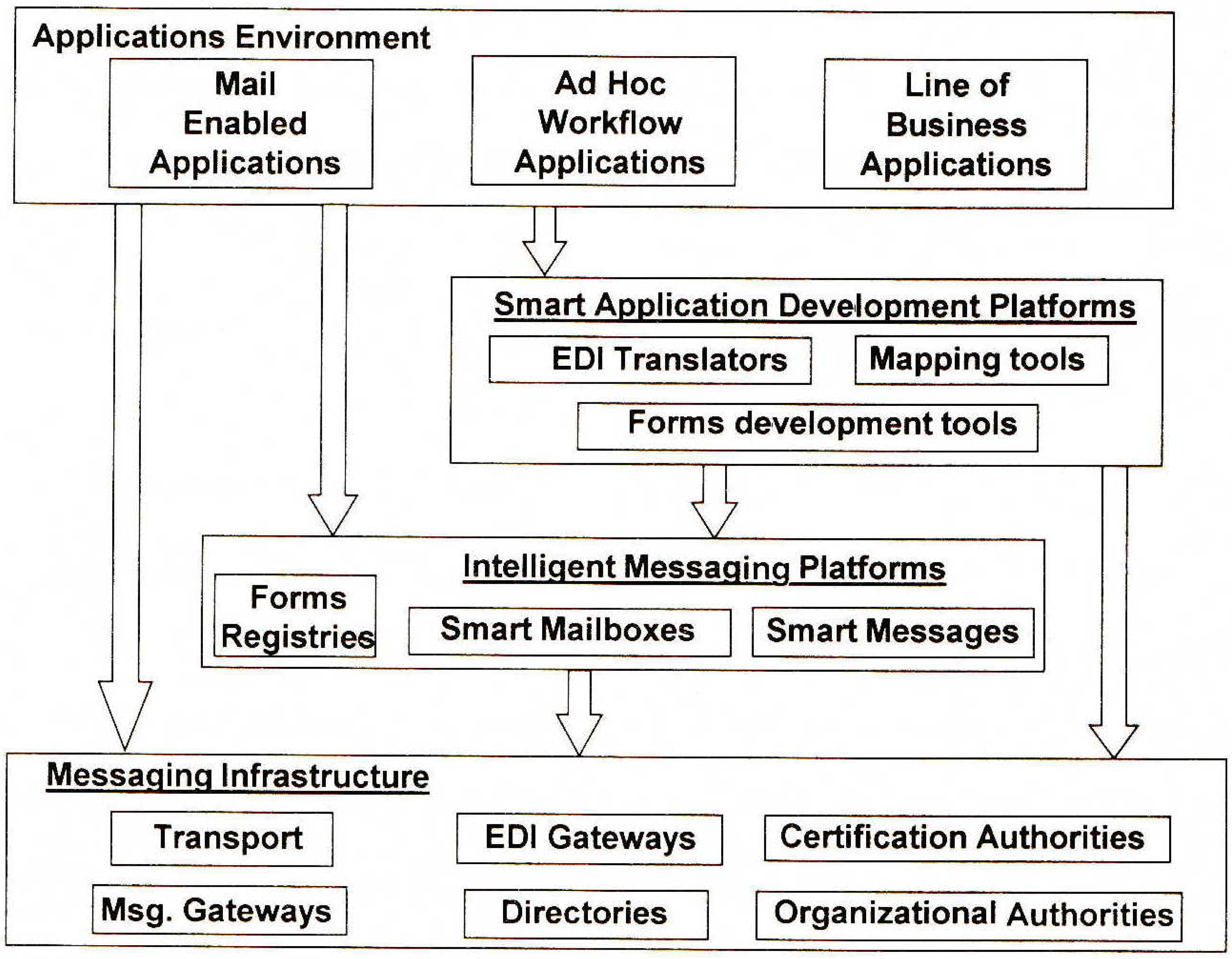


Figure 3: Integrated Electronic Commerce/Messaging Architecture Framework

These developments are exciting as new technology; but their true value will be in the way they assist in redefining the nature of our work and our economy. By providing speed and accuracy through the application of rules and intelligence, they will enhance business efficiency and reduce the drudgery and repetition that is often a part of blue and white collar work environments. Most significantly, they will lend new meaning to the challenge of what we call the "E-mail Frontier."

References

[1] Hammer, Michael, "Re-engineering Work: Don't Automate, Obliterate," *Harvard Business Review,* July-August, 1990.

[2] Ousterhout, John, "Using *Tcl* and *Tk* for Active Messages," Conference Presentation: E-mail World, November 1993.

[3] White, James E. "Telescript Technology: The Foundation for the Electronic Marketplace," General Magic, 1994.

[4] "Workflow specialist speaks out," *Network World,* July 5, 1993, pp. 27 and 31.

[5] Lee, Eugene; O'Neil, Denis; White, Thomas E.; & Spies Michael, "Workflow: Process Re-engineering," Conference Presentation: Electronic Mail Association, June 1993.

[6] D. Crocker, "Standard for the format of ARPA Internet text messages," RFC 822, August 1982.

[7] J. Postel, "Simple Mail Transfer Protocol," RFC 821, August 1982.

[8] D. Crocker, "Back to Basics: Internet Electronic Mail," *ConneXions,* Volume 9, No. 1, January 1995.

[9] M. Rose, *The Internet Message: Closing the Book with Electronic Mail,* Prentice-Hall, 1993.

[10] *ConneXions,* Special Issue: Electronic Messaging and Directory Service, Volume 6, No. 9, September 1992.

[11] John K. Ousterhout, *Tcl and the Tk Toolkit,* Addison-Wesley Professional Computing Series, ISBN 0-201-63337-X, 1994.

DANIEL J. BLUM has been active in open networking research and development since the early 1980s. He is well-known as a writer, educator, and consultant on integrated messaging and directory services. He has worked with key standards development committees, implemented messaging and directory systems, conducted extensive surveys of industry products, and advised industry and government users on the planning, architecture, procurement, and deployment of modern networks and value added messaging services based on multiple technologies including LAN email, workflow, X.400 backbones, Internet Mail, security, X.500 directories, and EDI. He has spoken at numerous trade conferences and his articles appear regularly in industry publications. E-mail: `dblum@access.digex.net`

DAVID M. LITWACK holds a doctorate in linguistics from Boston University. He has spent nearly twenty years in computer and data communications in the U.S. and abroad including stints in France, Algeria, Iran, Chad, and the Ivory Coast. He is currently president of dml Associates, a consultancy specializing in advanced telecommunications technologies, and an associate vice president of The Atlantic Group which provides educational programs in telephony and advanced intelligent networking. Dr. Litwack is author of numerous technical studies, product reports, education courses, and trade journal articles; and he is co-author with Dan Blum of *The E-mail Frontier.* E-mail: `0004623076@mcimail.com`

The World-Wide Web: How Servers Work

by Mark Handley & Jon Crowcroft, University College London

Introduction

A *World-Wide Web* (WWW) server listens on a TCP port (usually port 80) for incoming connections from clients. It expects a connecting client to speak a protocol called HTTP or *HyperText Transfer Protocol.* The connecting client is usually a browser such as *Mosaic,* which will request some information from the server, and the server will then return the requested information to the client (subject to the client passing any security restrictions the server may have).

HTTP is a pretty simple protocol. If you want to see what actually happens, you can *telnet* to a WWW server and talk to it yourself. (This can often be used to attempt to debug a misbehaving server. However, some servers don't check their input too carefully, and it may be possible to crash them by typing incorrect HTTP commands). The simplest HTTP request is `GET`. An example of telnetting to a server and issuing a `GET` request is:

```
telnet> open macpb1.cs.ucl.ac.uk 80
Connected to macpb1.cs.ucl.ac.uk
Escape character is '^]'.
telnet> GET /index.html HTTP/1.0
HTTP/1.0 200 OK
MIME-Version: 1.0
Server: MacHTTP
Content-type: text/html

<title>Mark's PowerBook on the Web</title>
<h1>Welcome to Mark's WWW server</h1>
This temporary server is running on an
Apple Macintosh PowerBook 180 using MacHTTP 1.3.
There's not much here right now, except for the
<a href=Default.html>HTTP documentation</a>.
```

The request we made was "`GET /index.html`" and additionally told the server we spoke "`HTTP/1.0`." The server responded with the document "`index.html`," and with additional information. The first line of the response says that the server is also speaking "`HTTP/1.0`," that the status code my request returned was "`200`," which in human terms means "`OK`." The next line gives information about the version of MIME. Then there's a line that says what type of server this was. And finally there's a line that says the "`Content-Type`" is "`text/html`." This last line is actually giving the MIME content type, which is how the server tells the client what to do with the information that follows. In this case it says that what follows is actually "text" (as opposed to an image, video, audio or a whole host of other possibilities), and that this particular text is in "`html`" format. If we'd asked for this information using a WWW client instead of *telnet,* the client would have read the `Content-Type` line, and known to feed the data following into its HTML interpreter.

MIME

MIME stands for *Multipurpose Internet Mail Extensions,* and was originally designed for sending multimedia electronic mail. The two main things it does are specify in a standard way what type of media the contents of a message actually are, and what form they've been encoded in for transmission. When Tim Berners-Lee was originally designing what would go on to become the World-Wide Web, he had exactly this same requirement—he needed a server to be able to specify to a client what a response contained and how it had been encoded. The e-mail people had got there first, and had already specified MIME, so there was no need to re-invent the wheel.

Content Types

MIME Content Types consist of a type (such as "`text`") and a subtype (such as "`html`"). The most common MIME types relevant to the WWW are:

- A "`text`" Content-Type, which is used to represent textual information in a number of character sets and formatted text description languages in a standardised manner. The two most likely subtypes are:
  - `text/plain`: text with no special formatting requirements.
  - `text/html`: text with embedded HTML commands
- An "`application`" Content-Type, which is used to transmit application data or binary data. Two frequently used subtypes are:
  - `application/binary`: the data is in some unknown binary format, such as the results of a file transfer.
  - `application/postscript`: the data is in the *PostScript* language, and should be fed to a *PostScript* interpreter.
- An "`image`" Content-Type, for transmitting still image (picture) data. There are many possible subtypes, but the ones used most often on the Web are:
  - `image/gif`: an image in the GIF format.
  - `image/xbm`: an image in the X Bitmap format.
  - `image/jpeg`: an image in the JPEG format.
- An "`audio`" Content-Type, for transmitting audio or voice data.
  - `audio/basic`: the data consists of 8KHz 8-bit mu-law audio samples.
- A "`video`" Content-Type, for transmitting video or moving image data, possibly with audio as part of the composite video data format.
  - `video/mpeg`: the data is MPEG format video
  - `video/quicktime`: the data is QuickTime format video

Suffixes, servers and MIME types

Now we know how a server tells a client what type of information is being returned, but how does the server figure out this information?

In the UNIX and DOS world, files are usually identified using file name suffixes. A file called `london_zoo.gif` is likely to be an image in the GIF format. Servers typically have a set of built in suffixes that they assume denote particular content types. They also let you specify the content types of your own suffixes in case you have any local oddities, or something new that the server designer hadn't thought of.

WWW servers generally reside on machines with a file system (a proxy server need not have a file system, but most do). The server's job is to make part of that file system publicly available by responding to HTTP requests. Its job is also to prevent the private parts of that file system from becoming public. Most file systems can be thought of as a form of tree, and *Universal Resource Locators* (URLs) also use this model: `http://www.cs.ucl.ac.uk/misc/uk/london.html` specifies the file called `london.html` which is in a directory called `uk`, which in turn is in a directory called `misc`. `misc` is also a directory, and it resides in the top level directory of the tree, which is sometimes simply called "/" (pronounced "slash").

continued on next page

How World-Wide Web Servers Work *(continued)*

URLs and server file systems

When the URL above specifies `/misc/uk/london.html`, this does *not* usually mean that the `misc` directory is really situated in the root directory of the entire file system. Instead it is situated in the root directory of the *subtree* that the server makes public. Any documents situated in this subtree are accessible to the server, and directories that are not in this subtree are not accessible. See Figure 1.

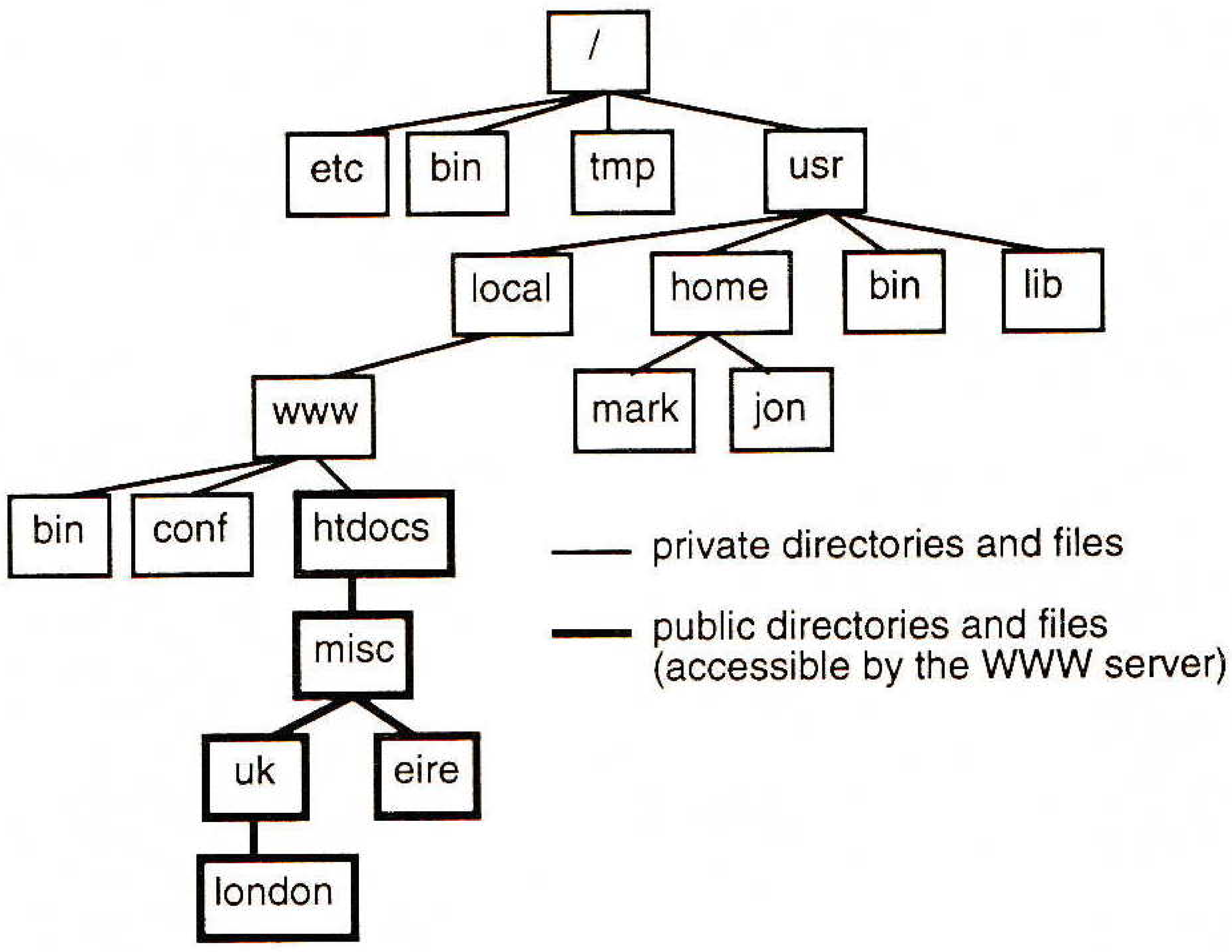


Figure 1: A WWW server makes a subtree of the filesystem public

However, most servers also allow you to provide some form of access control to files and subdirectories of the visible subtree. This protection can take the form of restrictions on which machines or networks a client can access a file from, or it may take the form of password protection. Which mechanisms a server provides depends on which server you choose, and we'll discuss a few of the better servers later in this article.

Another issue is raised where a server is running on a machine in a large multi-user environment such as a university. For instance, each student in a university can write files to their own filestore, but not anywhere else. However, we'd like our students to be able to create their own WWW pages, despite not having access to the WWW server's default public tree. Unix servers usually make available files placed in a special directory in the user's home directory. On NCSA and CERN servers, this directory is called "`public_html`" by default. So `http://www.euphoric-state-uni.edu/~janet/research/index.html` would map to `/usr/home/janet/public_html/research/index.html` in the filesystem shown in Figure 2.

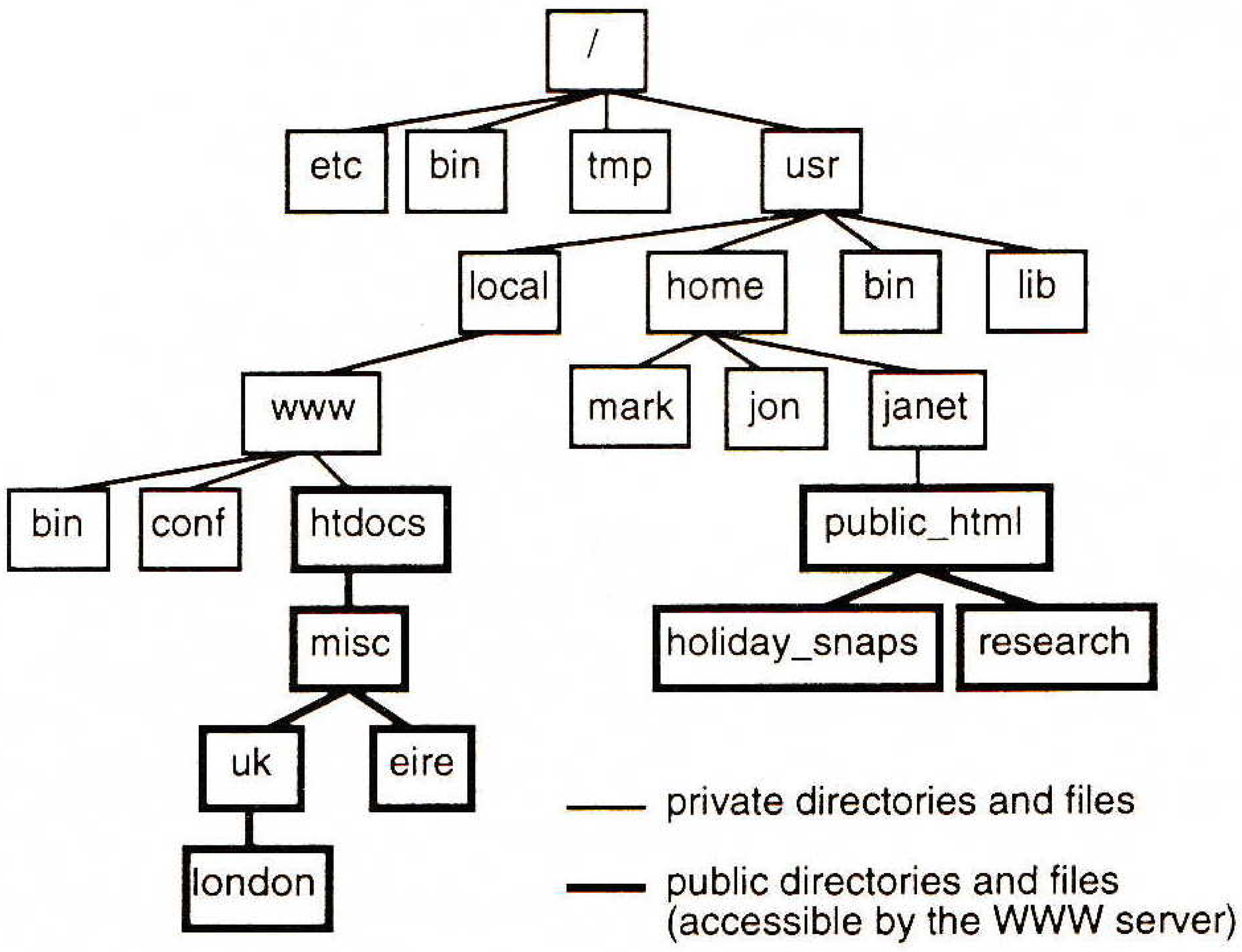


Figure 2: A WWW server exporting a user's `public_html` pages

Once we start to allow the WWW server access to areas of our file-system which can be modified by users that we don't necessarily trust, a whole set of security issues are raised. For instance, Unix allows symbolic links from one place in the directory tree to another to give the impression that files or directories are someplace else (Macs call symbolic links "Aliases"). Letting the server follow links can be useful, but it also can create problems. Just because a file is readable by other users on your own system does not necessarily mean it should be readable by users in other sites or countries!

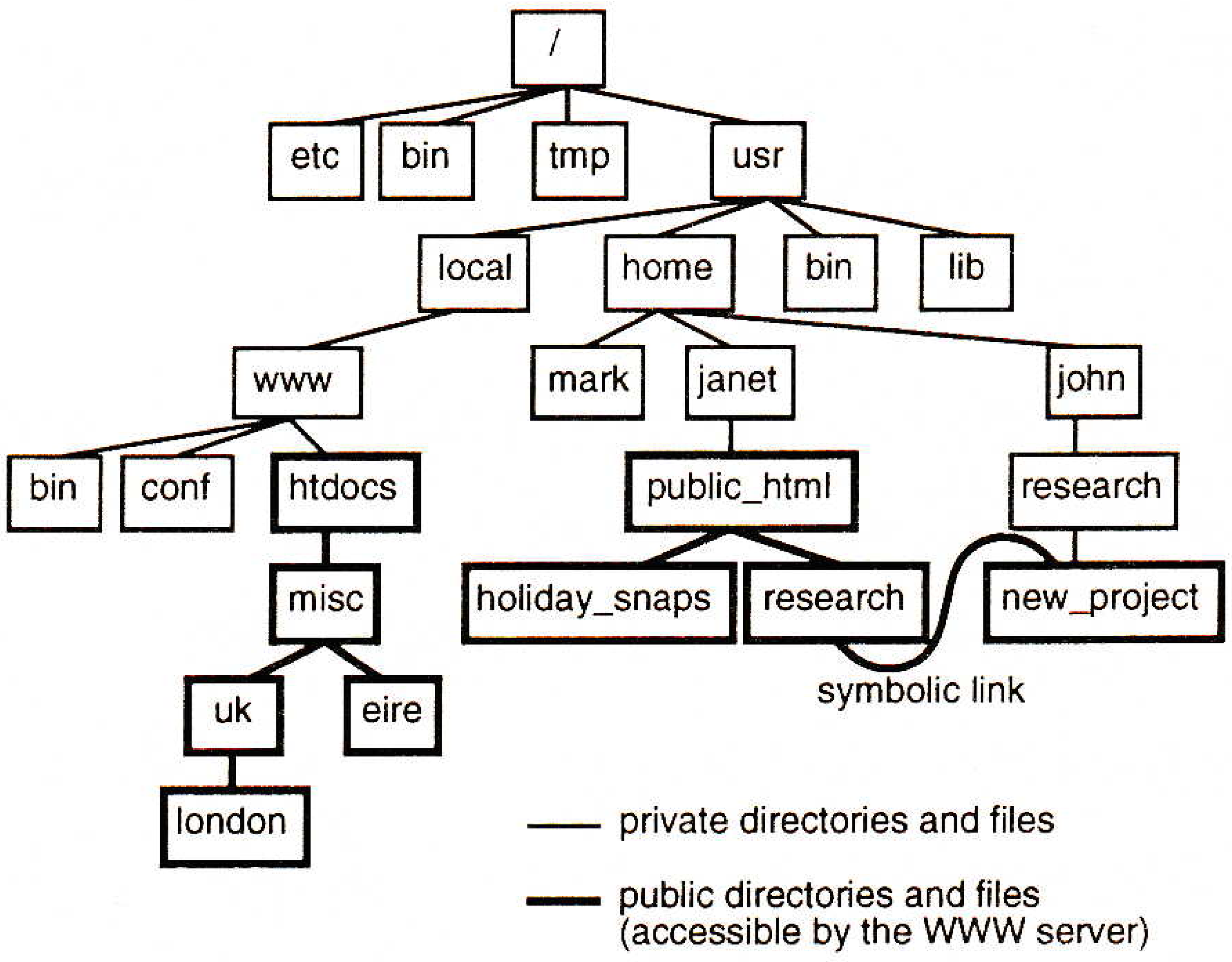


Figure 3: Problems with a WWW server following symbolic links

In Figure 3, we see that Janet has made a symbolic link from inside her "`public_html`" subtree to John's "`new_project`" directory making it accessible to the whole world without John's knowledge. Most servers allow different security options to be specified on a per subtree basis, and in this case, if following symbolic links had been switched off for "`public_html`" directories, the problem would have been avoided. MacHTTP simply prohibits the following of links (or Aliases as they're called on the Mac), which is another way to solve the problem.

Server scripts

The ability to define new programs to be run in the server when a request is made is what really makes the Web flexible and fun. An example is an active map, where a user clicks on a map, and the place they clicked is sent to the server along with their request. The server then runs a program or script which figures out where those coordinates apply to, and, depending on where the user clicked, it sends them the relevant next page of information. Another example is Cambridge University's coffee machine—they have a video camera pointed at the coffee pot, and a server script captures a picture of it using a video framegrabber, and sends the image to you so that you can see whether there's any coffee ready.

A standard called CGI or *Common Gateway Interface* has emerged for the writing of server scripts, and is supported by most servers. This means that scripts written for one server should be easily ported to another server.

Available servers

There are many WWW servers available, and more seem to be released each month. CERN's "List of Available Servers" is available with the URL:

`http://info.cern.ch/hypertext/WWW/Daemon/Overview.html`

CERN HTTPD

The CERN HTTPD server is probably the most fully featured WWW server. It supports much the same range of features as NCSA's server, with the addition of acting as a caching proxy server.

continued on next page

How World-Wide Web Servers Work *(continued)*

Proxy servers

If you have used a WWW client such as *Mosaic*, you have probably already used a proxy client. *Mosaic* and other clients built upon LibWWW can contact servers for protocols such as FTP and Gopher, and then convert the output of such servers into HTML for formatting and display on your screen (see Figure 4).

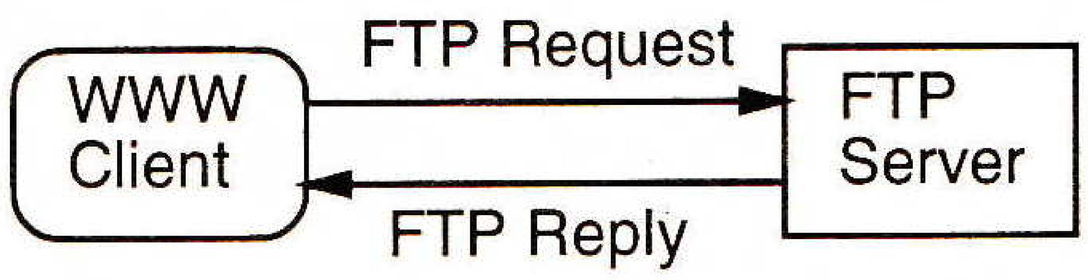


Figure 4: A WWW Proxy Client contacting an FTP Server

Proxy servers take this one step further—instead of your client contacting remote servers directly, your client makes an HTTP request to a proxy server. The proxy server then contacts the relevant FTP or Gopher server, and converts the results to HTML, before transferring them back to your client (see Figure 5).

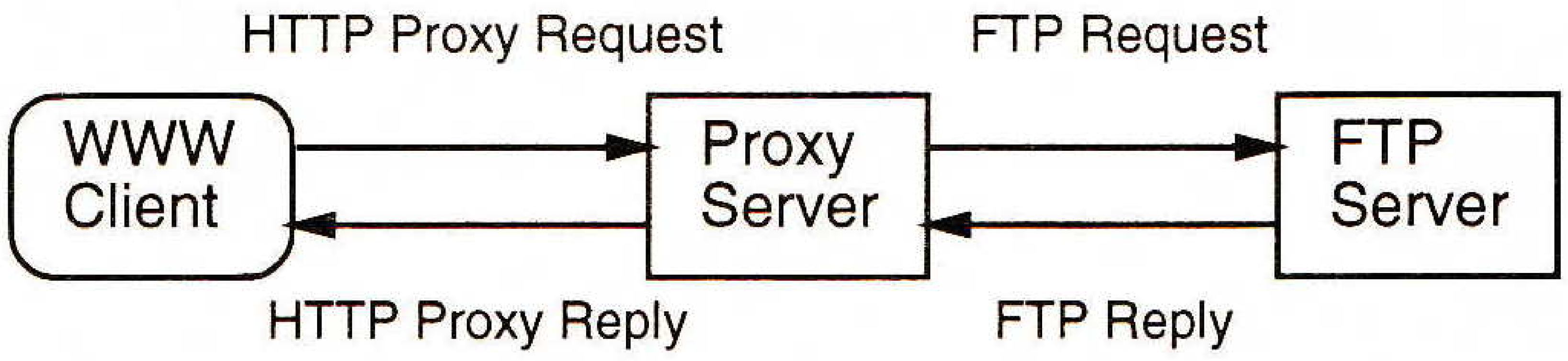


Figure 5: A FTP Proxy Server answering and HTTP request

A proxy server can also make connections to remote HTTP servers. At first glance, this wouldn't appear to benefit you, as the proxy then performs no conversion functionality, but it provides a way to provide network services to machines on a secure subnet without those hosts having to have direct access to the outside world. Thus secure sites can run a proxy server on their firewall machine, or SOCKSify only their proxy server without needing to modify the WWW client programs for all their different architectures (see Figure 6).

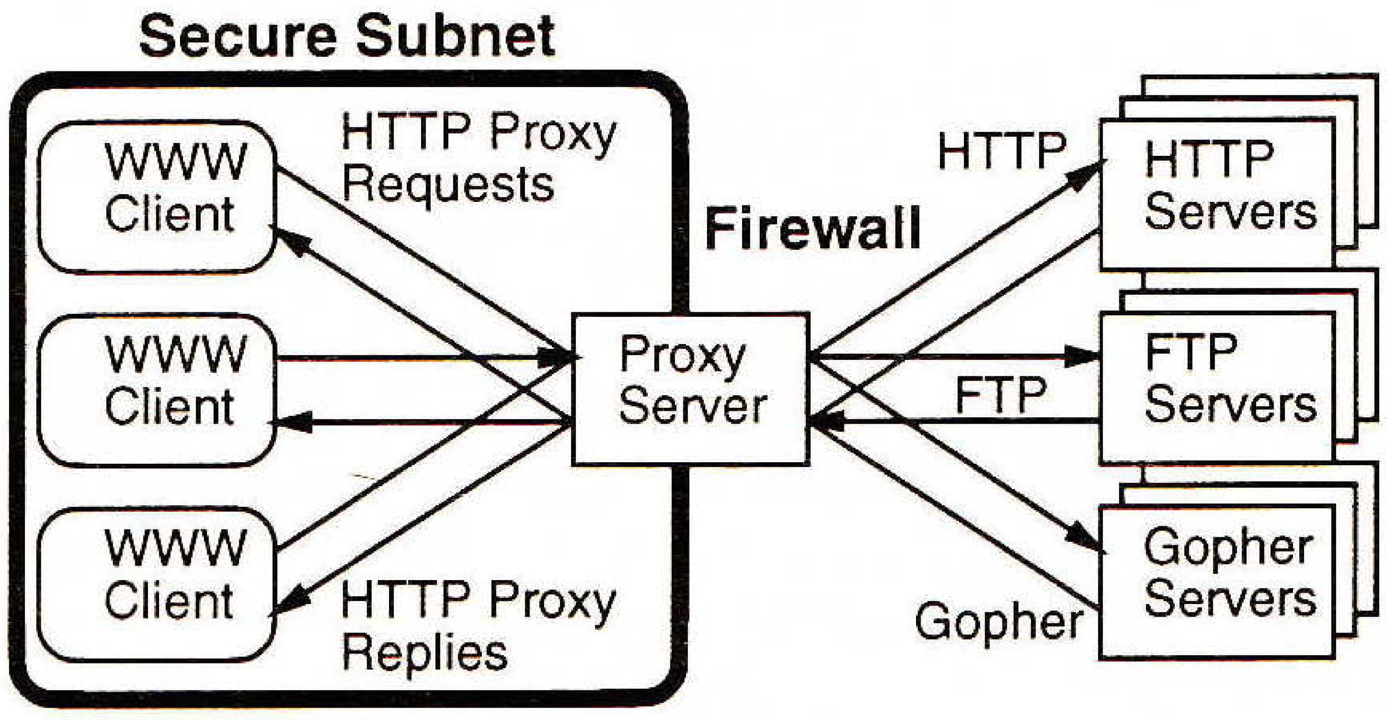


Figure 6: A Proxy Server on a Firewall

Even if you do not need this level of security, CERN's HTTPD can also provide caching facilities for clients using the server as a proxy. Caching facilities in the World-Wide Web are currently in their infancy, as many servers do not return expiry date information with documents, so deciding how long data should be cached before going back to look at the original is not a clear cut issue. However, CERN's server uses whatever information is available to it to make a decision about cache timeouts, and although it doesn't *always* do the right thing, it does substantially improve performance for frequently accessed pages, and most of the time it gets it right. (See Figure 7).

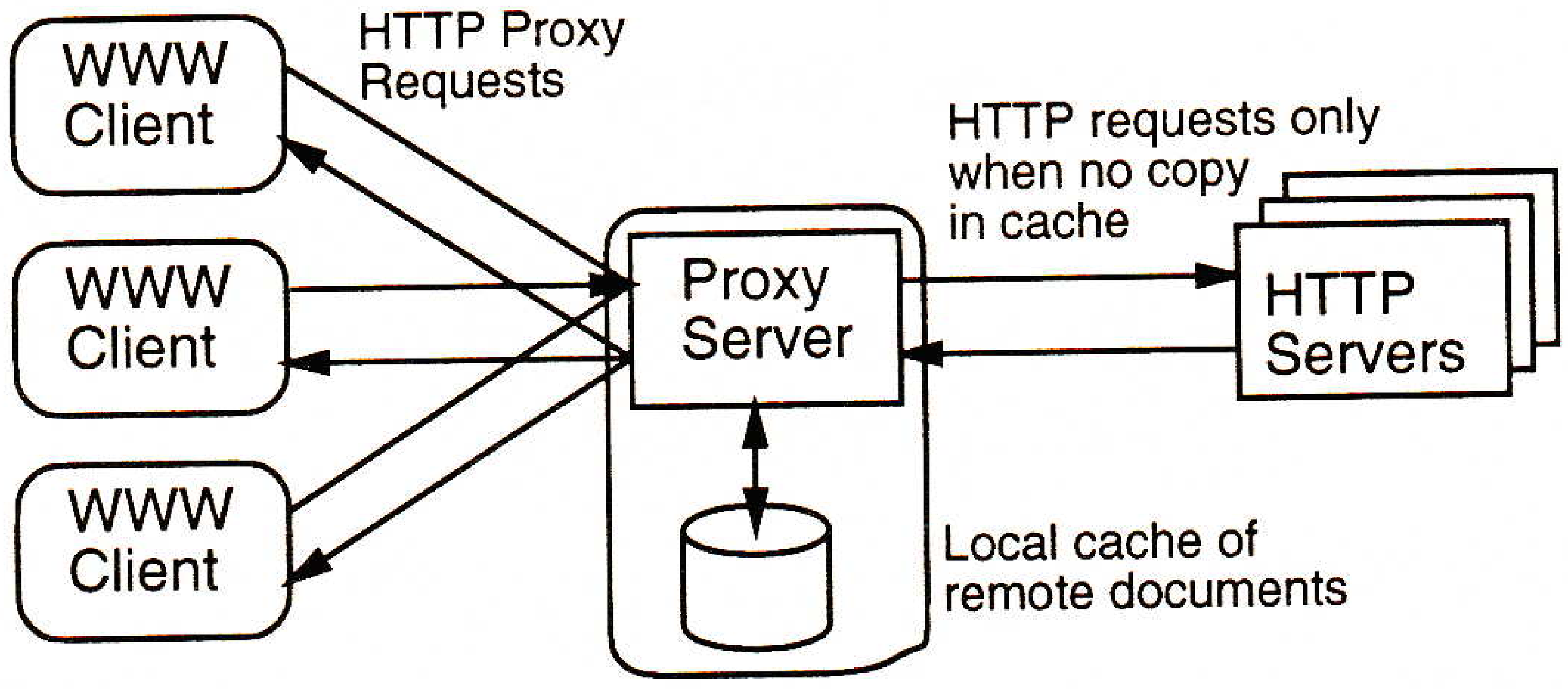


Figure 7: A Caching Proxy Server

CERN HTTPD configuration

The CERN HTTPD requires a single configuration file to function. By default, CERN HTTPD looks for this file as "`/etc/httpd.conf`," but it can be held elsewhere and the server told where it is using the `-r` command line flag.

The list of configuration options that CERN HTTPD supports is very extensive, and we encourage you to read the document *CERN HTTPD Reference Manual*. Most of the default options are fine to get you started.

Security

The CERN HTTPD server has a fairly sophisticated set of security features that can be enabled. Basically, they fall into three categories:

- Restricting hosts that can access areas of the server.
- Restricting users that can access areas of the server
- Restricting access to individual files

Common Gateway Interface

Before we had *Common Gateway Interface* (CGI), each server passed the query information into a script in its own way. Unfortunately this made it difficult to write gateways that would work on more than one type of server, so a few of the server developers got together and CGI was the result. Some servers don't yet support CGI, but most of the popular ones now do. CGI passes the information a script needs into the script in environment variables. The most important two are:

- `QUERY_STRING`: The server will put the part of the URL after the first "?" in `QUERY_STRING`
- `PATH_INFO`: The server will put the part of the path name after the script name in `PATH_INFO`

For instance, if we sent a request to the server with the URL:

```
http://www.cs.ucl.ac.uk/cgi-bin/htimage/usr/www/img/uk_map?404,451
```

and we had `cgi-bin` configured as a scripts directory, the server would run the script called `htimage`. It would then pass the remaining path information "`/usr/www/img/uk_map`" to `htimage` in the `PATH_INFO` environment variable, and pass "`404,451`" in the `QUERY_STRING` variable. In this case, `htimage` is a script for implementing active maps supplied with the CERN HTTPD.

The server expects the script program to produce some output on its standard output. It first expects to see a short MIME header, followed by a blank line, and then any other output the script wants returned to the client.

continued on next page

How World-Wide Web Servers Work *(continued)*

The MIME header must have one or more of the following directives:

- `Content-Type`: *type/subtype*. This specifies the form of any output that follows.
- `Location`: *URL*. This specifies that the client should request the given URL rather than display the output. This is a redirection. Some servers may allow the URL to be a short URL specifying only the file name and path—in this case the server will usually return the relevant file directly to the client, rather than sending a redirection.

The short MIME header can optionally contain a number of other MIME header fields, which will also be checked by the server which will add any missing fields before passing the combined reply to the client.

Under some circumstances, the script may want to create the *entire* MIME header itself. For instance, you may want to do this if you want to specify expiry dates or status codes yourself, and don't need the server to parse your header and insert any missing fields. In this case, both the CERN and NCSA servers recognise scripts whose name begins "`nph-`" as having a *"no parse header,"* and will not modify the reply at all. Under these circumstances your script will need access to extra information to be able to fill out all the header fields correctly, and so this information is also available via CGI environment variables.

Handling active maps

One nice feature which is now supported by most graphical WWW clients is the ISMAP active map command, which can be associated with an HTML inline image. This tells the WWW client to supply the x and y coordinates of the point the user clicks on within the image. For example, this HTML tells the client this image is an active map:

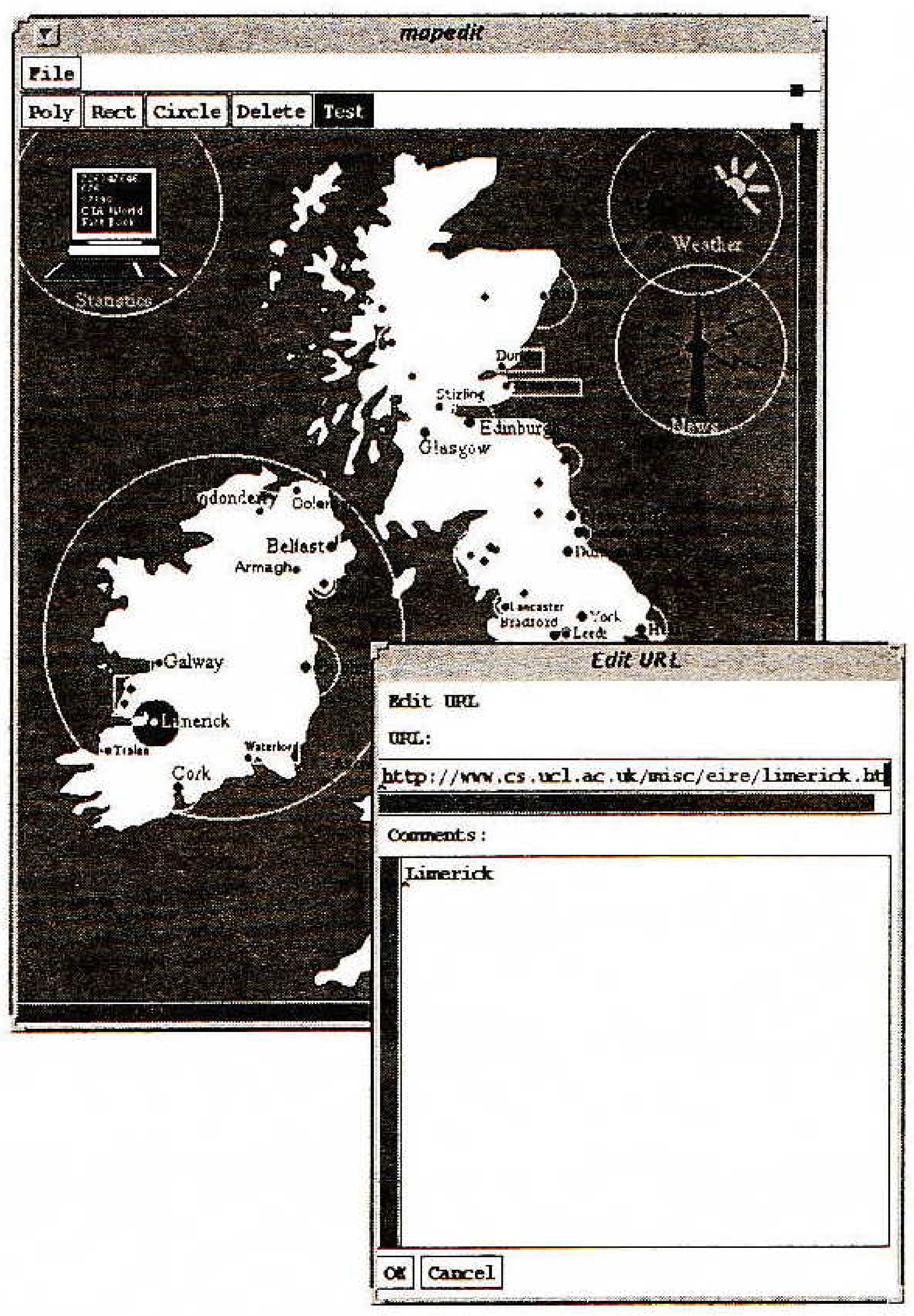
```
<a href=/cgi-bin/imagemap/uk_map>
  <img src=uk_map_lbl.gif ISMAP>
</a>
```

when the user clicks on the map at, say, point (404,451), her client will submit a GET request to the server:

```
GET /cgi-bin/imagemap/uk_map?404,451
```

For this to do anything interesting, the server must interpret "`/cgi-bin/imagemap/uk_map`" as something special—a command to be executed rather than a file to be retrieved. How the server decides this is a command depends on the type of server, but whichever server you run, the "404,451" part will then be passed to the command as parameters.

When the command is executed, it could generate output that is to be returned directly to the client—for instance the command could generate HTML directly as output. However the usual way imagemaps are used is to access other existing pages of HTML using HTTP redirection. This is where the server first returns to the client the URL of the place to look for the page corresponding to the place they clicked on the map, and then the client goes and requests this new URL (usually without bothering to ask the user).



Mapedit is an example of a tool used to create an active map.

Handling forms

Forms are one way the World-Wide Web allows users to submit information to servers. All the mechanisms described so far allow users to choose from a set of available options. Forms let the user type information into their Web browser and then get the server to run a program with their submission as input. Examples of things you might type are keys to search a database (e.g., what films was Zazu Pits in?).

HTML provides a number of commands for telling the client to do something special. The first command is `FORM` which tells the client that everything between one `<FORM>` command and the next `</FORM>` terminator is part of the same form. The form command can take a number of attributes:

- `ACTION=http://www.host.name/cgi-bin/query`: This gives the URL of the script to run when the form is submitted. You must supply an `ACTION` attribute with the `FORM` command.
- `METHOD=GET`: This is the default method for submitting a form. The contents of the form will be added to the end of the URL that is sent to the server.
- `METHOD=POST`: The post method causes the information contained in the form to be sent to the server in the body of the request.
- `ENCTYPE=application/x-www-form-urlencoded`: This specifies how the information the user typed into the form should be encoded. Currently only the default, "`application/x-www-form-urlencoded`," is allowed.

If your server supports the `POST` method, it is advisable to use it, as if you use the `GET` method, it's possible that long forms will be truncated when they're passed from the server to the script.

The INPUT command

Now you have an empty form, you probably want to provide some boxes and buttons that the user can set. These are created using the INPUT tag. This is used in a similar way to the IMG tag for images—there's no need for a terminating tag as it doesn't surround anything. There are several types of INPUT tag, denoted by the TYPE attribute:

- `<INPUT TYPE=text NAME=users_name>`: This is a simple text entry field that we've called "`users_name`." The user never sees this `NAME` attribute displayed on her client—it is purely so we can keep track of which field is which when we come to process the form. Text entries also allow you to specify:
  - `VALUE="enter your name here"`: This lets you specify the default text to appear in the entry box.
  - `SIZE=60,3`: This lets you specify the size of the entry box is characters. For example the above says the entry box should be 60 characters wide and three lines high.
  - `MAXLENGTH=8`: This lets you specify the maximum number of characters you'll allow to be entered in a *single line* text entry box. For instance, you might only allow a user to enter eight characters as their user name.
- `<INPUT TYPE=password NAME=users_passwd>`: This is also a text entry field, but the characters the user types are displayed as stars so that other people can't read the password from their screen. Password fields also support the `VALUE`, `SIZE` and `MAXLENGTH` attributes.

continued on next page

How World-Wide Web Servers Work *(continued)*

- **<INPUT TYPE=checkbox NAME=veggie>**: This is a single button which is either on or off. Checkboxes also support the following attributes:
  - **VALUE="true"**: This is the value to return if the checkbox is set to "on." If it's set to "off," no value is returned.
  - **CHECKED**: This says that the checkbox is "on" by default.
- **<INPUT TYPE=radio NAME=food_style VALUE=indian>**
  <INPUT TYPE=radio NAME=food_style VALUE=chinese>
  These are a collection of buttons. Radio buttons with the same name are grouped together so that selecting one of them turns the others off like the channel tuning buttons on some radios. Radio buttons also support the **VALUE** and **CHECKED** attributes, but only one radio button can be specified as **CHECKED**.
- **<INPUT TYPE=submit VALUE="Press Me to Submit">**: This is a button that submits the contents of the form to the server using the method in the surrounding **FORM**. Submit buttons don't have a **NAME** attribute, but you can specify the label for the button using a **VALUE** attribute.
- **<INPUT TYPE=reset VALUE="Press Me to Start Again">**: This is a button that causes the various boxes and buttons in the form to reset to their default values. Reset buttons also don't have a **NAME** attribute, and allow a **VALUE** attribute to label the button.

The SELECT command

If you want to provide the user with a long list of items to choose from, it's not very natural to use radio buttons, so HTML provides another command—**SELECT**. Unlike **INPUT**, this does have a closing **</SELECT>** tag. Each option within the list is denoted using the **<OPTION>** tag. Options must be plain text—no embedded HTML commands are allowed:

```
<SELECT NAME="food style">
<OPTION> Chinese
<OPTION> South Indian
<OPTION> North Indian
<OPTION> Greek
</SELECT>
```

SELECT must have a **NAME** attribute, and also allows the following attributes:

- **SIZE=3**: This says how many of the options are visible at once, in this case, three.
- **MULTIPLE**: This allows the user to select more than one item from the list. The default is that only one item can be selected at once.

OPTION tags can also have a **SELECTED** attribute which says that this option is selected by default. If the **SELECT** command has a **MULTIPLE** attribute, then several **OPTION** tags are allowed to be pre-selected in this way.

The TEXTAREA command

If you want to allow the user to enter a large amount of text, you could use an **<INPUT TYPE=text>** tag, but HTML also provides another command—**TEXTAREA. TEXTAREA** fields automatically have scroll bars on *Mosaic,* and any amount of text can be entered into them.

TEXTAREA fields must have a **NAME** attribute, and also must have **ROWS** and **COLS** attributes specifying how large the visible area of the **TEXTAREA** is in characters. **TEXTAREA** fields, like **SELECT** fields must have a closing tag:

```
<TEXTAREA NAME="address" ROWS=4 COLS=60>
Any default contents go here
</TEXTAREA>
```

The default contents must be ordinary text with no HTML formatting commands.

An example form

An example of form that demonstrates many of these features is:

```
<HEAD><TITLE>Pub List Feedback</TITLE></HEAD>
<BODY>
<H1>Pub List Feedback</H1>
Please use this form to let us know about any good pubs you
come across in London.
<FORM ACTION=http://www.cs.ucl.ac.uk/cgi-bin/pubform METHOD=POST>
<HR>
<B>Pub Name:</B>
<INPUT TYPE=text NAME=pubname SIZE=40>  <P>
<B>Pub Address:</B>
<INPUT TYPE=text NAME=pubaddress SIZE=40,4>
<P>
<B>Area of London:</B>
<SELECT NAME=area SIZE=4>
<OPTION SELECTED>Bloomsbury
<OPTION>Theatreland
<OPTION>The City
<OPTION>Kensington and Chelsea
<OPTION>Out of the Centre
<OPTION>Further afield
</SELECT>
<HR> <TEXTAREA NAME=description ROWS=6 COLS=80>Describe your pub here!
</TEXTAREA>
<p>
<INPUT TYPE=radio NAME=grade VALUE=1>Average.
<INPUT TYPE=radio NAME=grade VALUE=2 CHECKED>Worth going to.
<INPUT TYPE=radio NAME=grade VALUE=3>Worth a detour.
<INPUT TYPE=radio NAME=grade VALUE=4>Worth a long detour!
<HR>
<B>Your Name:</B>
<INPUT TYPE=text NAME=username SIZE=40>
<P>
<B>Your email address:</B>
<INPUT TYPE=text NAME=useremail SIZE=40>
<P>
<I>(You can leave these blank if you don't want to be credited)</I>
<HR>
<INPUT TYPE=submit VALUE="I've finished now">
<INPUT TYPE=reset VALUE="Ooops, can I start again?">
</FORM>
</BODY>
```

Figure 8 on the next page shows how this looks on the NCSA *Mosaic* browser.

Submitting forms to the server

As we mentioned above, there are two ways you can submit a form to a server—the **GET** method and the **POST** method. These methods refer to the type of request the browser makes to the client.

continued on next page

How World-Wide Web Servers Work *(continued)*

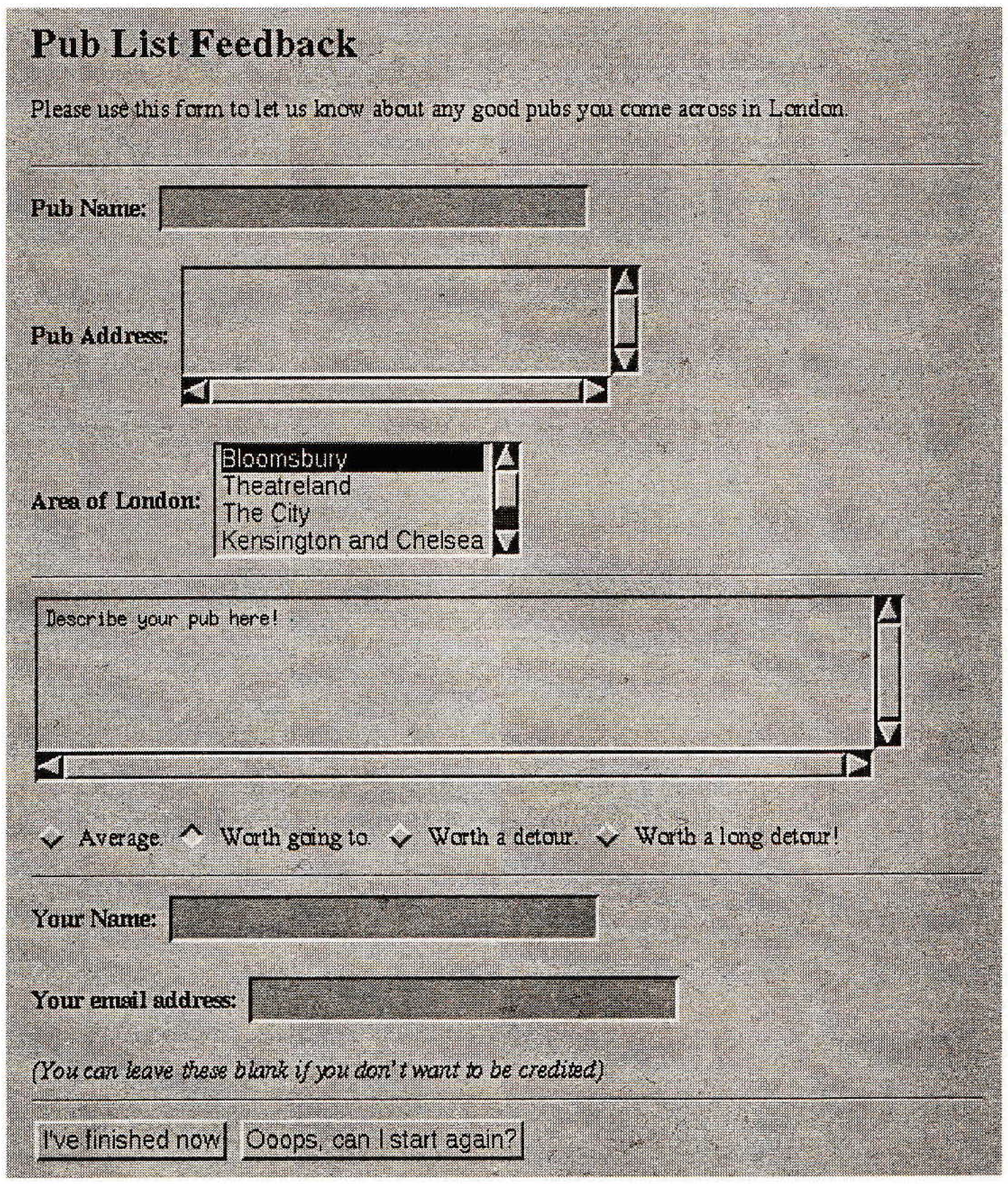


Figure 8: An example form as displayed by NCSA *Mosaic*

GET is the normal way a browser requests a page or an image from a server—the URL gives the location of the file we want or the name of the script that will produce the data we want. We saw with image-maps that we could also send a small amount of additional data to a script in the URL by putting it at the end of the URL after a question mark. You can also send all the information contained in your form to the server in the same way. However, it's possible that some servers may have size limits in the amount of data they can pass to the script in this way.

POST is an entirely different method from those we've seen so far. A **POST** request consists of a URL which refers to the script we wish to run, and then a URL encoded body which contains the data from our form. (URL encoding replaces spaces with "+" and encodes other special characters as "%XX" where XX is the ASCII code for the character in octal).

If your server supports the **POST** method, it is recommended to use it —using the **GET** method may work, but if your forms are large, it is really stretching the intended purpose of **GET** somewhat.

The cgiparse approach

The CERN HTTPD server comes with a useful script called **cgiparse**, which does most of the hard work. It will work with either the **GET** or **POST** methods, though again we recommend using **POST** for forms of any length.

cgiparse reads the **QUERY_STRING** environment variable (as set if you use the **GET** method) or, if **QUERY_STRING** is not set, reads **CONTENT_LENGTH** characters from standard input (as set if you use the **POST** method). What it does next depends on which flag you give it, but for now we're only interested in the "**-form**" flag.

"`cgiparse -form`" outputs a string which, when evaluated by the Bourne shell, sets environment variables (with "`FORM_`" prepended to their names) for each of the form elements. It also URL decodes the variables for you. (The Bourne shell is the standard command interpreter on Unix systems.) Thus a script to do the same task as above (now written in Bourne shell!) would be:

```
#!/bin/sh
eval 'cgiparse -form'
$filename=$FORM_pubname
$doc_root="/cs/research/www/www/htdocs"
$fullfilename=$doc_root"/misc/uk/london/pubs/auto-"$filename".html"

#Write the entry to a file in HTML
echo "<TITLE>"$FORM_pubname"</TITLE>" > $fullfilename
echo "<H1>"$FORM_pubname"</H1><HR>" >> $fullfilename
echo "<I>"$FORM_pubaddress"</I><P>" >> $fullfilename
echo "<h2>Area:</h2> "$FORM_area"\n" >> $fullfilename
echo -n "<h2>Description:</H2>" >> $fullfilename
.....and so on...
```

`cgiparse` can take many other command line options to modify its behaviour, and can be used for tasks other than form processing—we recommend the *CERN httpd Reference Manual* which is available on the Web for a more detailed explanation.

Server performance issues

Whilst this article should give you enough information to know what the main issues are with setting up and running a WWW server, it is still possible to end up with a server that doesn't perform too well. If your server is lightly loaded, then this is probably not a problem, so ignore this section. However, if you provide interesting information, your server may be inundated with requests, and you may need to consider its performance. If you're used to systems administration, then the following tips may be obvious, but many servers are run by people with little systems administration experience, and as this group of people is increasing rapidly, we've added a few pointers:

- *How does your server start?* WWW servers running on Unix (or similar) systems have two ways to start—either from `inetd` (the Internet Daemon) when a connection arrives or at system boot time and the application listens continuously for incoming connections. Starting from `inetd` is much slower than starting a single copy at system boot time.

- *Where are the files you're serving actually stored?* Although you may consider you LAN to be fast and your WAN to be slow, if your server has to get your files from a fileserver on a different machine, this will load your LAN unnecessarily, and slow down your server's response time. On some systems, it may be possible to use a caching filesystem to do this if you can't put all the files on the WWW server machine.

- *How does your server resolve its UID and GID?* Both the CERN and NCSA servers let you start the server as root (so it can use port 80), and then set the user ID and group ID to a none privileged account. This is a very good idea from a security point of view. However, if your server uses NIS (*Network Information System*) to obtain the user and group IDs (particularly the supplementary groups), it will have to pull your user and group database files across your LAN from the NIS server for every WWW access. It may be a good idea to run a slave NIS server on the same machine as the WWW server to improve this performance.

continued on next page

How World-Wide Web Servers Work *(continued)*

• *Does your server have enough memory?* It's pretty obvious, but if a machine is thrashing (the active processes use more memory than is available, so you're paging from disk continuously), then it will perform badly. On systems such as the Macintosh, there's only one running server, but the memory it uses increases as the number of simultaneous connections increases. Check the server at busy times to ensure its memory allocation is sufficient.

• *How efficient are your CGI scripts?* If your server is seeing a lot of accesses to CGI scripts, those scripts may dominate the performance of your server. Scripts that do a lot of database searching, or use a lot of files from another fileserver may cause you problems. If this appears to be a problem, consider running another WWW server on the database machine itself, moving the files to the WWW server, or using a caching filesystem to improve the situation.

• *How stable are your CGI scripts?* It's not uncommon for badly written CGI scripts to contain bugs that cause them to loop indefinitely. Whilst you would probably notice this immediately with a normal program, a WWW user not getting a response from a server is quite likely to try again, and hit the same bug again! Some servers can be configured to kill CGI scripts after a certain amount of time. Check how long you think your scripts should take to run, and set this value accordingly. However, if you do this, you may never discover malfunctioning scripts unless you set them up to write a log entry when they're started and remove the log when they finish correctly.

• *Have you turned off RFC 1633 authentication?* Some servers (such as CERN HTTPD) permit you to enable RFC 1633 authentication, which attempts to query the client machine about the username of the person making the request. This information is unreliable at best, and if your server is busy, it is best to disable this feature.

• *Who are your users?* You can analyse who your users are (or rather where they come from). If many of your users come from one area that is not local to you, consider setting up copies of your data elsewhere.

• *Have you considered setting up a caching server?* If your server is busy with requests from elsewhere, your local users will get poor performance to the rest of the world. Consider setting up a caching server (such as CERN HTTPD), and configuring your local user's clients to use it. This will help them, and will also reduce the load your own users cause elsewhere. If you can't set up your own caching server (because there's not one available for your hardware), then consider using someone else's. Enquire whether your network provider is running a caching server, and if they're not then encourage them to do so.

Most of this is common sense if you know a little about how the WWW server actually works. If you have access to any network monitoring facilities, it can be enlightening to look at how much local network access your server is doing when responding to a request. If it's more than you expect, then the server is probably using resources from another local server (such as a fileserver or NIS server), and if it is excessive, this will be detrimental to performance.

Where else to look

All the popular WWW Client programs have self-contained help systems, which are usually just hotlists or bookmarks that point to the distribution sites. Another place to look if you prefer the printed page, is a forthcoming book by the authors of this article, entitled *World Wide Web: Beneath the Surf,* to appear in March 1995.

JON CROWCROFT is a senior lecturer in the Department of Computer Science, University College London, where he is responsible for a number of European and US funded research projects in Multi-media Communications. He has been working in these areas for over 14 years. He graduated in Physics from Trinity College, Cambridge University in 1979, and gained his MSc in Computing in 1981, and PhD in 1993. He is a member of the ACM, the British Computer Society and the IEE. He was general chair for the ACM SIGCOMM 94 symposium. He is also on the editorial teams for the *Transactions on Networks* and the *Journal of Internetworking.* E-mail: `J.Crowcroft@cs.ucl.ac.uk`

MARK HANDLEY graduated from UCL with a 1st class honours degree in Computer Science with Electrical Engineering in 1988. As a PhD student at UCL, he studied novel neural network models and their visualisation. Since 1992, he has been a Research Fellow, working on the RACE CAR project and on MICE, now managing the UCL part of MICE. His current research interests include Multimedia Systems, especially audio and video encoding and compression, Distributed and Heterogeneous Systems, and HCI and graphics. E-mail: `M.Handley@cs.ucl.ac.uk`

This article is based on material in Mark Handley and Jon Crowcroft's, *The World Wide Web: Beneath the Surf,* ISBN 1-85728-435-6, UCL Press. —Ed.

Call for Papers

The 20th Annual IEEE *Conference on Local Computer Networks* (LCN) will be held October 15–18 1995 in Minneapolis, Minnesota.

Theme

The emphasis on this year's conference is on practical experience using computer networks in both the local area and in the global area. This unique approach simulates a workshop environment and allows for an effective interchange among users, researchers, and vendors. Some of the primary goals of the conference are to enable those involved in the local computer network field to share experiences, lessons learned, and prototype data and analysis. Because of these objectives, papers based on experience are especially solicited. The focus of the 20th LCN will be "Practical Experience Using and Deploying Local Computer Networks." Papers that cover these areas are explicitly sought and will be given preference.

Information for authors

All authors must submit 5 full copies of the full technical paper by mail or delivery service. *Do not submit complete papers by fax.* The first page must contain: title of the paper, author's names including affiliations, complete mailing address, telephone and fax numbers, Internet or BITNET address, and a 250 word (maximum) abstract (double-spaced) in English.

Topics

- Internetworking/Routers/Bridges
- High Performance Protocols
- Multimedia
- Metropolitan Area Networks
- Distributed Applications
- LAN/MAN/WAN Integration
- Wide Area Networks
- Standards
- ATM
- Network Management
- Fibre Channel Networking
- Remote Monitoring
- High Speed Networks
- Wireless Networks
- Error Control Techniques
- Emerging Technologies
- Congestion Control
- Realtime Networks
- FDDI and FDDI-II
- AI Networks

Send papers to

Joe Bumblis, Program Chair
3M
3M Center, Building 224-4N-27
St. Paul, MN 55144-1000
USA
Phone: +1 612-737-4763
Fax: +1 612-736-7689
E-mail: `jrbumblis@mmm.com`

Important dates

Submissions due:	March 21, 1995
Notification of Acceptance:	June 27, 1995
Camera-ready copy due:	August 1, 1995

Call for Papers

The 1995 IFIP International Working Conference on *Upper Layer Protocols, Architectures and Applications* (ULPAA) will be held in Sydney, Australia December 11 through 15th, 1995. The event is held under the auspices of IFIP Technical Committee 6 IFIP Working Group 6.5 and is hosted by University of Technology, Sydney and the Australian Computer Society.

Overview

The Internet is now progressing into a "marketplace" of worldwide interworking services. The TCP/IP protocol suite is simple and can be used by an unlimited variety of applications. It is accessible worldwide by more than 30 million users and continues to grow. Message Handling is still the most popular service and, although widely used, still contains considerable technical and social problems. However, the Internet already offers much richer applications such as the *World-Wide Web,* privacy and integrity services, and information shopping. The Internet puts us in the fascinating situation of engineering this huge and complex communication system while we use it at the same time. The conference is a place where active researchers can exchange their understanding, experience, and design plans about open network cooperation technology.

Increasingly, the effective use of computers depends upon smooth interworking between disparate and diverse systems, communicating with each other using a wide variety of networking technologies. Such smooth interworking, in turn, depends critically on standardized communication protocols, which are themselves dependent on clearly-understood and well-specified architectures for distributed applications. As new applications are developed, new protocols are vital to the success of the applications in the world. The ULPAA conference provides a pre-standards forum where leading researchers can discuss promising and problematic developments in the world of distributed applications. Past conferences in this series already have had significant impact on the earliest stages of standard development in both the ISO and Internet protocol suites. We are soliciting papers that will help to focus the efforts of the networking community and to point out new directions for continued progress.

Topics

Application architectures:

- Implementation, and experience with distributed applications
- Models and designs
- Programming environments
- Group communication models and services
- The impact of human factors on upper layer protocol design and implementation
- Multimedia applications and communications
- Management and operation of distributed services
- The perspective of new applications like:
  - World-Wide Web (WWW)
  - Directory services
  - Privacy and integrity services (PEM and PGP)
  - Information shopping
  - Electronic publication
  - Business on open networks

Impact on applications of underlying services:

- Interconnection of upper layer and application entities
- Mobile communications
- Upper layer network management and naming
- Universal resource naming schemes
- Presentation and session layer issues
- Security and privacy provision
- Very high-speed networking

Standards:

- Upper layer conformance and interoperability testing activities
- The role of the standardization process for upper layer protocols

Format

The purpose of the conference is to provide an international forum for the exchange of information on the technical, economic, and social impacts and experiences with upper layer protocols, architectures and distributed applications. The conference format will be two and a half days of conference paper presentations combined with one half a day of workshops.

Submission Guidelines

Prospective authors are invited to submit unpublished original contributions (not exceeding 5000 words) which describe recent research results or developments directly relevant to upper layer protocols, architectures or distributed applications. IFIP WG 6.5 intends to use modern communication technology for its cooperation within the WG. The subject of our research shall be the basis of our daily work. Therefore, we are already using electronic mail for all WG business including discussions, event preparations, joint reviewing, etc. The next step will be to work on electronic publication for the conference proceedings. For this purpose, electronic submission of papers is highly recommended.

Publication

Accepted papers will be published by Chapman and Hall, London, the general publisher for IFIP. A preprint of the proceedings will be provided to attendees. We will thoroughly examine the possibility of electronic publication in accordance with IFIP and the Publisher.

Where to send papers

Please submit either five paper copies of your paper or one file in electronic form (plain text and/or in *PostScript* format) to one of the program committee co-chairs:

Dr. Nathaniel S. Borenstein
First Virtual Holdings, Inc.
25 Washington Avenue
Morristown, NJ 07960
USA
E-mail: `nsb@nsb.fv.com`
Tel: +1 201 993 8586
Fax: +1 201 993 3032

Dr. Ruediger Grimm
Gesellschaft fuer Mathematik und Datenverarbeitung GmbH
P.O. Box 100138
D-64201 Darmstadt
GERMANY
E-mail: `grimm@darmstadt.gmd.de`
Tel: +49 6151 869 716
Fax: +49 6151 869 785

continued on next page

Call for Papers *(continued)*

Prof. Dr. Bob Kummerfeld
University of Sydney
Computer Science Department
NSW 2006
AUSTRALIA
E-mail: `bob@cs.su.oz.au`
Tel: +61 2 692 3423
Fax: +61 2 692 3838

Tutorials

On Monday and Tuesday, December 11–12, time and space is reserved for a number of tutorials. We invite interested persons to submit proposals for tutorials. If you have ideas about a tutorial and wish to discuss these ideas informally, please just send e-mail to one of the program chairs. The following topics are being considered:

- Upper Layer/Application Layer Architecture
- Security Models, Mechanisms and Systems
- Message Handling Harmonization
- Directory Services Harmonization
- PGP and PEM
- Network Management
- ASN.1
- Coexistence and Transition to OSI Applications
- Distributed Application Programming Environments
- Multimedia Internet Mail (MIME)
- World Wide Web Applications
- Payment and Purchase Systems on the Internet

HIPPARCH Workshop

The HIPPARCH Workshop will be meeting at the same location, parallel to the tutorials in the two days 11th and 12th of December 1995, prior to the conference proper. ULPAA participants are welcome to attend the workshop. The workshop is organized by UTS within the context of the HIPPARCH Basic Research Action project (European-Australian collaboration), sponsored by CEC DG XIII. The objective of the HIPPARCH project is to study and design high performance communication architectures and implementations, based particularly on the "Application Level Framing" and "Integrated Layer Processing" concepts, and with an emphasis on automatic protocol compilation techniques. Several research groups engaged in this area will be represented. The first HIPPARCH workshop was held at INRIA in December 1994. This is the second workshop, which will present the outcomes of the project in workshop format.

Important dates

Today:	Send a message, letter, phone, or fax to either of the contacts above stating your intention to submit a paper, or stating your general interest in the conference.
May 15 '95:	Full version of papers due for review.
July 15 '95:	Notification of acceptance/rejection.
Nov. 1 '95:	Camera-ready papers required for publication.

More information

More information can be found at:

```
http://www.ee.uts.edu.au/ifip/ULPAA95.html
http://www.darmstadt.gmd.de/TKT/IFIP6.5/IFIP-WG6.5.html
http://www.iaik.tu-graz.ac.at/IFIP/tc6_home.html
```

N+I Online!
The Networking Trade Show on the Internet

by Samantha White, SOFTBANK Expos

Introduction

N+I Online! is the first implementation of a Networking Trade Show on the Internet by SOFTBANK Exposition and Conference Company. It is an on-line, interactive extension to the traditional physical trade show, free to the networking community two weeks before, during, and two weeks after the physical NetWorld+Interop event. N+I Online! allows people to "attend" this virtual trade show via the World-Wide Web ("the Web") on the Internet.

N+I Online! uses the latest technologies on the Web to bring together buyer and seller more effectively than ever before. N+I Online! has several powerful features, including a high-powered search engine from Verity Inc., customized software programs from Internet Media Services that register and track on-line attendees, and *Virtual Places* Technology from Ubique, the only technology to allow for real-time voice or text conferencing as well as guided tours over a Web page.

Fundamentally, the objective of the SOFTBANK's 21st Century Trade Show is to deliver essential information about technology, products, vendors, users and customers. We provide the access and delivery of this information, and connect information with the right customer. The concept includes three basic, but significant components: education, an interactive show floor (exhibits, InteropNet, hands-on demonstrations), and subsequent communication via publications. N+I Online! is a natural extension that extends the impact. Now attendees can preplan their agendas at the event electronically, and seamlessly tap into related research and evaluations of the products and technologies highlighted at the exhibition and conference.

The Internet and the World-Wide Web

Today, growth of the networking industry is explosive, changing the way we do business. There are over 1.3 million hosts in the commercial `.com` domain—the equivalent of "virtual storefront addresses" on the Internet; an increase from 9,000 in 1991. The Internet's population has grown from an estimated 1 million individuals in 1988 to more than 30 million today, with hundreds of thousands joining each month. The Internet knows no geographical boundaries. From 4.85 million computers around the world today, the Internet is expected to increase to more than 100 million machines in five years. Already, the Internet reaches over 77 countries. The greatest area of growth on the Internet is the *World-Wide Web*.

The World-Wide Web (WWW), started by the European Laboratory for Particle Physics, (CERN) located in Geneva, Switzerland, represents an ever-increasing mesh of global networked information linked by a common body of software, protocols, and conventions. There are currently more than 7,000 host computers that work together as one to provide WWW service. These Web servers provide access to documents (formatted by the HyperText Markup Language [HTML]) and hypertext links to other documents or files on the network.

N+I Online!

Unlike raw World-Wide Web resources, SOFTBANK Expos provides a new model for the logical organization of information that is specific to the current interests and requirements of the networking industry, just as they do by assembling the hottest topics in networking under one roof at the NetWorld+Interop events. Attendees have access to the complete view of the state of networking, all in one location—be it physical or on-line.

continued on next page

N+I Online! *(continued)*

How it works

N+I Online! consists of:

- Exhibitor Virtual Booths
- On-line Attendee Access
- Technology Pavilions
- Easy search methods for finding exhibitor & product information
- On-line conference info and N+I Conference Planner program
- Special on-line events

N+I Online! is available for two weeks before the physical show (the Preview), the week of the show (the Premier), and two weeks after the show (the Encore). During Preview, attendees can register for N+I Online! and establish an N+I Online! account.

Attendee services

At NetWorld+Interop Las Vegas, there will be "Network Application Centers" with full Internet access. Attendees can visit all N+I Online! exhibitors, get last minute conference updates and booth changes, evaluate products and services in conjunction with their team members at the home office, and create a summary of their itinerary based on the results of their exploration in N+I Online! During the Encore, attendees can review exhibitor's products, e-mail requests for additional information, or participate in bulletin-board discussion groups. Exhibitors can leverage this opportunity to follow-up with attendees and close the sales they began at the physical event.

N+I Online! attendees can access exhibitor, product and conference information in order to plan their NetWorld+Interop itinerary and participate in interactive events. N+I Online! provide exhibitors with a focused Internet gathering spot for networking professionals, multiple ways to be found by targeted buyers, a direct channel of communication with those buyers, and detailed lead retrieval information.

The N+I Conference Planner is in an interactive way for attendees to build a conference itinerary. It allows attendees to search for Conference sessions by day, track, instructor, or a full-text search of the course descriptions. With the click of a mouse, attendees can create a "Conference Itinerary." Attendees can access their account at anytime to adjust, update, and printout their Conference Itinerary.

Attendees can also use N+I Online! to find specific products and exhibitors. Searches by (1) Technology Pavilions (a logical grouping of exhibitors by technology category, such as ATM), (2) Full Text Searches or (3) Alphabetical Listings are available. An attendee can locate an exhibitor offering a specific product and then use a search engine from Verity, Inc. to link with the page of information within the exhibitor's Virtual Booth. Attendees can communicate with exhibitors quickly via e-mail. Attendees and exhibitors using Ubique's *Virtual Places* technology can talk to each other in real time, and exhibitors will be able to guide attendees through their Virtual Booth over the Internet.

Exhibitor participation

There are three levels of N+I Online! participation for exhibitors: Virtual Booth, Virtual Booth Starter Kit, and Interactive Virtual Booth. From a novice without a Web site to companies with highly developed Web sites, N+I Online! has a solution for every exhibitor. Exhibitors may select one of the following types of participation:

• *Virtual Booth:* Like concrete booth space on the physical floor of NetWorld+Interop, this is an exhibitor's foundation for reaching the community of networking professionals. A Virtual Booth is a direct link from the N+I Online! server to an exhibitor's self-appointed Home Page URL.

NETWORLD+INTEROP® 95 N+I Online!

THE NETWORKING TRADE SHOW ON THE INTERNET

The content of an exhibitor's booth is the responsibility of the exhibitor. Exhibitors may have the N+I Online! server point to a vendors corporate Home Page, or they may design specific pages to compel and inform the N+I Online! qualified buyer.

• *Virtual Booth Starter Kit:* For companies without a Web site, SOFTBANK Expos will provide templates for the easy creation of a Virtual Booth, and host that Virtual Booth on the N+I Online! server for the duration of N+I Online! SOFTBANK Expos will also provide a Site Usage Report of virtual attendee's exploration of exhibitor's Virtual Booth, at the close of N+I Online!

• *Interactive Virtual Booth:* In addition to the direct link and benefits provided to Virtual Booth participants as described above, SOFTBANK Expos offers Virtual Places™ Technology by Ubique, which enables interactive experiences for the exhibitor and attendee.Virtual Places software include:

- Real-time voice and text chat interaction during meetings, attendee visits, or special on-line events.
- Guided Tours, where exhibitors can travel throughout the Internet with multiple attendees on board a "Virtual Tour Vehicle" designed by the exhibitor.
- On-line Business Cards™, which provide customer profile information at the click of a mouse, are accessible to exhibitors during interactions with potential buyers. This business card information is deposited and recorded on an exhibitor's server.
- Active Mail™ allows exhibitors to invite attendees who have the Sesame client to visit their virtual booth. When attendees accept the invitation, the Sesame program automatically transports the on-line attendee to the exhibitor's Web page.

Partners

SOFTBANK Expos has a license agreement with Ubique to distribute their Virtual Places family of products.

Internet Media Services (IMS) and SOFTBANK Expos have jointly developed the framework for N+I Online! IMS developed software for the foundation of N+I Online! Additionally, IMS has created the Virtual Booth Starter Kits for Exhibitors new to the World-Wide Web.

Verity is a provider of document search and retrieval software and are adapting their premium product to work with Web pages (HTML documents). Verity creates a keyword index of every document on a server, so that an attendee can type in keywords, and have the search engine return the appropriate Web link. The user then clicks on the result and gets transported to the appropriate Web page. Verity is providing the search engine for N+I Online!, with unlimited user usage. SOFTBANK will distribute a free "spider" to all on-line exhibitors. The spider will simply index the local system and returns the completed index to a central location to create a huge virtual index. A central index will be hosted on the SOFTBANK server so that when an attendee comes in to look for a product, they retrieve a list of links, click on a link and go right to the source, allowing them to "drill straight down," rather than having to browse paths.

Debut

N+I Online! opens March 13th, 1995. You can access N+I Online! by pointing your Web browser to the URL: `http://www.interop.com`

CONNEXIONS
303 Vintage Park Drive
Suite 201
Foster City, CA 94404-1138
Phone: 415-578-6900
FAX: 415-525-0194

FIRST CLASS MAIL
U.S. POSTAGE
PAID
SAN JOSE, CA
PERMIT NO. 1

ADDRESS CORRECTION
REQUESTED